

Symbols

!path ./myclasses 284
(X)aaS 20
@BeforeClass annotation 267
@BeforeSuite annotation 267
@BeforeTest annotation 267
@DataProvider annotation 259
@Delegate annotation 303
@Test annotation 261, 268
|script| row 285
|start| command 287
\$HOME/.m2/settings.xml 147
\$M2_HOME/conf/
settings.xml. 124
\$M2_REPO/com/huetter-
mann/myartifact 126

Numerics

1.0.2-SNAPSHOT
dependency 155

A

acceptance style 314
acceptance test-driven develop-
ment. *See* ATDD
acceptance testing 73, 254–255
application 272–273
frameworks 272–283
gluing tests and processing
documents 276–278
running tests 278–283
specification 273–276
accidental complexity 27

ACID (Atomicity, Consistency,
Isolation, and Durabi-
lity) 173
ActionFixture class 274
activating tasks 104
Active Directory authentica-
tion 143
add() method 258
additional Bamboo Builds
tab 108
admin screen, JIRA 100
advanced permission set-
tings 140
aggregation, inheritance
and 127–128
Agile 15–17
development 98, 102, 303
strategies 34–57
and project manage-
ment 36–40
CI 43–50
component repository 50–51
process pitfall 53–57
productive work-
spaces 42–43
quality, standards, and rele-
ase cycles 51–53
version control and single
coding stream 40–41
Agile ALM
approach 194, 201, 227, 250
definition 6
fundamentals 6
key benefits 14
Agile ALM infrastructure 96
Agile ALM summary 4
Agile ALM system 96

Agile application lifecycle man-
agement. *See* ALM
Agile ecosystem 35
Agile Manifesto 39
Agile menu 102
Agile Planner tool 105
Agile practices, with Groovy
language 298–306
DSLs 302–303
Groovy Maven eco-
system 305–306
low ceremony and script-
ing 301–302
testing with easyb and Spock
frameworks 303–305
Agile project management 99
Agile teams 99
Agile-Trac plug-in 111
agnostic regarding tools 316
Airport class 293
AirportEditor 294
AirportSteps classes 293–294
ALM (Agile application lifecycle
management) 3–33
building blocks of 17–26
configuration, customiza-
tion, and plug-
ins 22–23
open source culture 24
polyglot program-
ming 23–24
service orientation 19–20
stakeholder focus 17–19
task-based and outside-in
development 21–22
views on release manage-
ment 19

ALM (*continued*)
 description of 5–7
 evolution of software engineering 7–17
 Agile 15–17
 ALM 8–15
 SCM 8–13
 example use case 30–33
 with lightweight tooling 26–30
 ALM approach 104
 ALM environment 114
 Amazon Web Services. *See* AWS
 Ant approach 195
 Ant language, FTP communication with 195–199
 Ant tool, running tests with 278–280
 antipatterns 184–185
 AntRun plug-in 199
 Apache POI library 270
 API-breaking changes 306
 application containers, interfacing with Cargo tool 185–187
 applications
 for acceptance testing 272–273
 injecting build numbers into 227–229
 Archetype plug-in 177
 archetype project 177
 ArgumentCaptor 183
 ArgumentMatcher 183
 artifact bootstrapping 135
 artifact types 5, 298
 artifactId coordinate 143, 175–176
 artifactId data 133
 Artifactory build info 233
 Artifactory button 238
 Artifactory commercial version 142
 Artifactory Configuration section 230
 Artifactory plug-in 230
 Artifactory project 138, 141–142, 145
 Artifactory repository manager
 enterprise component repository 137–145
 staging artifacts in 232–236
 Artifactory server 229
 Artifactory UI 216, 229, 231
 artifacts. *See also* reference 14, 30, 55
 multiple, with assemblies 208–210

staging
 and promoting 236–238
 in Artifactory repository manager 232–236
 assemblies
 assembly command 167
 multiple web apps with 208–210
 Assembly plug-in 162
 assemblyProject 165
 AST Transformations 302
 ATDD (acceptance test-driven development) 254
 ational Unified Process 35
 Atlassian Confluence wiki 101, 109–110
 Atlassian Connector for Eclipse 107
 Atlassian tool 100
 atomic deployments, with Jenkins /Artifactory bridge 231–232
 atomic VCS commit 94
 Atomicity, Consistency, Isolation, and Durability. *See* ACID
 attaching context information 104
 auditing 218–225
 with Jenkins server 218–223
 with Sonar tool 223–225
 audits 55–56
 Auto-generated settings.xml files 141
 automated acceptance tests 83
 automated deployment process 18
 automatic property setting 91
 automatic testing 46–47
 automation 7, 14, 26, 71–72
 AWS (Amazon Web Services) 20

B

backup POM files 158
 balanced scorecard approach. *See* BCS
 Bamboo builds 106
 Bamboo documentation 109
 Bamboo integration in JIRA 108
 Bamboo plug-in 215
 Bamboo Release Management plug-in 109
 Bamboo server, build view with 107–109
 Bamboo web application 92, 97, 108
 barrier-free approach 11, 251, 255–256
 barrier-free approaches 316
 barrier-free development 298
 barrier-free experience 298
 barriers 4–5, 197
 bash script 195
 BCS (balanced scorecard) approach 21–22
 BDD (behavior-driven development) 22, 256–259, 306–315
 in FitNesse tool 283–296
 GivWenZen tool 285–290
 scenarios 293–296
 isolating systems with Mockito tool 182
 Scala language
 build ecosystem of 315
 specs2 library 307–312
 BDD artifacts 312
 BDD frameworks 303
 BDD style 296
 BDD tests 308
 BDDMockito 182
 Beck, Kent 35
 behavior-driven development. *See* BDD
 big bang integrations 48
 binaries, managing in conjunction with sources 135–137
 binary artifacts 51
 BitNami 111
 black box tests 252
 blame storming 77
 bottom level approach 16
 branch by abstraction approach 241
 branches, release 152–157
 branching strategies 136
 branching, for feature branch-driven CI 238–246
 breaking feature-branch CI 243
 breaking locks 90
 broken builds 52
 broken code 42
 Browse Source tab, Trac 115
 bug tracking, with Trac Tool 109–116
 installing 111–112
 Roadmap tab and tickets 115–116
 timeline and sources 114–115
 wiki 113
 bugfix branch 41
 bugs 42, 80

- Bugzilla 103
 - build agents, and cloud computing 205–207
 - Build Browser 238
 - build ecosystems, of Scala language 315
 - build fragments, running in Jenkins server 225–227
 - build lifecycle 128
 - build numbers, injecting into applications 227–229
 - build pipelines 123
 - build scripts 122
 - build section 154, 219
 - Build server job setups 243
 - build view, with Bamboo server 107–109
 - BUILD_NUMBER property 226
 - build-staging 45
 - build.name 235
 - build.number 145, 235
 - builds 31
 - .NET framework, using TeamCity tool to trigger 203–207
 - Bamboo 106
 - congruent, and workspace management 171–177
 - remote, with TeamCity integration server 187–190
 - using MSBuild system to build .NET framework projects 202–203
 - Builds panel 233
 - built-in Power Assert command 298
 - built-in project descriptor 163
 - built-in support, for JUnit 298
 - built-in testing 301
 - burn-down chart artifact 67
 - burn-down charts 99, 102
 - business values 17
 - business-facing tests 251
- C**
-
- CalculatorTest class 258
 - CalendarEditor class 290
 - calendars, release planning 84–85
 - Capability Maturity Model Integration. *See* CMMI
 - Captain Hooks 89
 - capture and replay. *See* CR
 - caret character 314
 - caretakers 78–79
 - Cargo Maven2 plug-in 185
 - Cargo tool, interfacing application containers with 185–187
 - central repository 41
 - central servers 42
 - ceremony, reducing 301–302
 - change control function 9
 - change enablers 57
 - changesets 14, 94–95, 114
 - check command 274
 - check-ins. *See* changesets
 - checkstyle 133
 - Checkstyle plug-in 221–223
 - Checkstyle tool 56
 - CI (continuous integration) 6, 18, 31, 43–50, 172
 - automatic testing and inspections 46–47
 - continuous delivery and deployment 47–48
 - Conway's law 48–49
 - description of 45–46
 - server 121–123
 - synchronization and continuous improvement 49–50
 - virtuous cycle 48
 - CI (continuous integration) tools 191–246
 - .NET framework 201–207
 - using MSBuild system to build projects 202–203
 - using TeamCity tool to trigger builds 203–207
 - Cobol platform 193–200
 - FTP communication 195–200
 - preconditions for 194–195
 - Cobol applications 192–194
 - Cobol artifacts 193, 198
 - Cobol development 193–194
 - Cobol platform 193–200
 - FTP communication
 - with Ant language 195–199
 - with Java language 199–200
 - preconditions for 194–195
 - Cobol sources 194–198, 200
 - Cobol support 194
 - CocaColaMachine class 309
 - code branches 46
 - code changes 105
 - code freeze 79–81
 - code freeze interval 136
 - code freeze phase 94
 - codebase 107
 - coding streams, single 40–41
 - Cohn, Mike 46
 - collaborations 6, 12, 34
 - collaborative development 297–317
 - Agile practices and polyglot programming with Groovy language 298–306
 - DSLs 302–303
 - Groovy Maven ecosystem 305–306
 - low ceremony and scripting 301–302
 - web apps for multiple environments 207–213
 - applying different configurations 210–212
 - multiple artifacts with assemblies 208–210
 - using distribution profile and executing example 212–213

- collaborative development (*continued*)
 - testing with easyb and Spock frameworks 303–305
 - BDD 306–315
 - Scala build ecosystem 315
 - with Scala language specs2 library 307–312
 - summary 316–317
 - collaborative testing 251–259
 - acceptance tests 254–255
 - BDD 256–259
 - data-driven tests 253–254
 - executable specifications 256
 - ubiquitous language 255–256
 - command-line commands 124
 - command-line tool 201
 - commercial add-ons 143
 - commit (put changes) 96
 - commit operation 88
 - commit-triggered feature-branch CI build 242
 - commitments, release 75–76
 - common-testsuite.xml file 279
 - Commons Net library 199
 - commons-logging 225
 - communication 35
 - communication points 137
 - compilation errors 42
 - compile scope 175
 - compile-time AST
 - Transformations 304
 - Compiled Cobol sources 194
 - complex multimodule settings 127
 - component repositories 50–51, 121, 132–150, 232
 - Artifactory repository manager 137–145
 - managing sources and binaries in conjunction 135–137
 - using subversion to serve 145–150
 - component teams 78–79
 - comprehensive documentation 39
 - Concurrent Versions System. *See* CVS
 - config.xml file 236
 - configurable relational database 141
 - configuration 22–23
 - configuration audit function 9
 - configuration environments, visual 203
 - configuration identification 55
 - configuration identification function 9
 - configuration items 8–9, 54, 56–57
 - configurations, applying different 210–212
 - Confluence plug-in 101
 - congruent build management 43, 124, 171–177
 - connection element 152
 - consecutive CI break 240
 - construction phase 63
 - containers, application
 - interfacing with Cargo tool 185–187
 - continuous build 45
 - continuous building jobs 242
 - continuous delivery 47–48
 - continuous improvement 7, 26, 49–50
 - continuous inspection 47
 - continuous integration. *See* CI
 - continuous risk management 16
 - contract negotiation 40
 - Control Objectives for Information and related Technology. *See* CobiT
 - convenient browsing 106
 - Conway's law 48–49
 - copy-everything execution
 - ID 166
 - copy-javadoc 166
 - copy-modify-merge model 90
 - copy-sources 166
 - corporate release calendar 85
 - CR (capture and replay) 253
 - created artifacts 14
 - createFlight method 288, 290
 - creating a baseline approach 51
 - CRM (customer relationship management) 19
 - cron-like syntax 214
 - cross-functional process 49
 - cross-functional team. *See* feature teams
 - cross-project knowledge 124
 - CruiseControl server 107
 - cryptographic key 161
 - cryptology, using with Maven tool 161–162
 - Currency object 310
 - customer collaboration 40
 - customer relationship management. *See* CRM
 - customer-centric acceptance tests 256
 - customization 22–23
 - customizing 75
 - CVS (Concurrent Versions System) 95, 105
 - cycle time 47
-
- ## D
-
- daily Scrum 76–77
 - dashboarding 45
 - dashboards, Jenkins server jobs and 215–218
 - data access 20
 - data definition languages. *See* DDLs
 - data manipulation language. *See* DML
 - data synchronization 13
 - data-aggregation support 218
 - data-driven testing 253–254, 259–272
 - testing web UI 264–269
 - TestNG framework and data-driven approach 259–262
 - with Microsoft Excel 270–272
 - with XStream library 262–264
 - database administrators. *See* DBAs
 - database elements 83
 - database owners. *See* DBO
 - DatabaseHistory 179
 - DBAs (database administrators) 18, 70
 - DBO (database owners) 70
 - DDLs (data definition languages) 70
 - death sprints 71
 - default Trac installation 113
 - DefaultSelenium class 267
 - defects 47
 - DelayFlightWithAirportTaxiTime Test 293
 - deliverable. *See* artifact
 - delivery slots 79–81
 - delivery, continuous 47–48
 - delta release 70
 - DeMarco, Tom 52
 - demilitarized zone. *See* DMZ
 - dependencies, POMs and 125–126
 - dependency injection, isolation and 178–180
 - DependencyJAR 163

dependency management 175
 dependency manager 127
 dependencySet element 166
 deploy standard lifecycle
 phase 134
 deployment 47–48
 deployments, atomic
 with Jenkins/Artifactory
 bridge 231–232
 descriptorRefs section 165
 descriptors element 165
 design phase 9
 DesignForExtensionCheck 221
 dev WAR file 210
 developer branches 137
 developer checks 105
 developerConnection section 152
 developers 17
 developing Cobol applications 194
 development
 behavior-driven
 isolating systems with Mockito tool 182
 collaborative. *See* collaborative development
 task-based. *See also* task-based development 21–22, 116
 test-driven, isolating systems
 with Mockito tool 180–182
 development activity 107
 development efforts 14
 development environment 170–190
 congruent builds and workspace management 171–177
 interfacing application containers with Cargo tool 185–187
 isolating systems with Mockito tool 177–185
 antipatterns 184–185
 behavior-driven development 182
 equals() method 182–183
 favorite static imports 182–183
 in test-driven development 180–182
 isolation and dependency injection 178–180
 organizing static imports 182–183

 remote builds with TeamCity integration server 187–190
 development phases 5, 9
 development process 123
 development team 16
 DevOps movement 18
 discipline 14
 distributed VCS 86
 distribution management
 servers 51
 distribution profiles 212–213
 distributionManagement
 section 128, 140, 147–148, 157–158
 DML (data manipulation language) 70
 DMZ (demilitarized zone) 133
 documentation 109, 150
 documents, processing 276–278
 domain language 255
 Domain-Driven Design 255
 domain-specific languages. *See* DSLs
 DSLs
 domainStep 288
 double maintenance problem 10
 dry-run feature 159
 DSDM 35
 DSLs (domain-specific languages) 301–303, 305, 311
 Dummy object 177
 Duvall, Paul M. 43

E

easyb framework, testing
 with 303–305
 EasyB stories 304
 EC2 (Elastic Cloud Computing) 20
 Eclipse bundle 127
 Eclipse editor 103
 Eclipse IDE 98
 Eclipse Maven classpath container 176
 Eclipse OSGi 127
 Eclipse plug-ins 103, 116, 131, 188
 Eclipse standard distribution 104
 Eclipse task view 103
 Eclipse TeamCity Remote Run dialog box 188
 Eclipse tool 103–105
 Eclipse-based task framework 92, 96, 98, 103

ecosystems
 Maven tool 131–132
 Maven, with Groovy language 305–306
 Scala language build 315
 effectiveness 54
 efficiency 54
 efficient comparison 107
 elaboration phase 63
 Elastic Cloud Computing. *See* EC2
 email alerts 106
 empire-building 49
 encapsulate data 20
 enforce-once goal 160
 Enforcer plug-in 161
 engineering activities 67
 enter command 273
 environment-specific application properties 207
 environment-specific configuration files 213
 environment-specific data 207
 environment-specific information 210
 environments, multiple 207–213
 equals() method 182–183, 310
 error-prone copying 175
 error-prone way 135
 errors, cost of repairing 44
 Excel documents 253
 Excel library 270
 Excel, Microsoft. *See* Microsoft Excel
 executable specifications 256
 expandproperties command 207
 extensions construct 220
 external DSLs 307
 external libraries 133
 Extreme Programming 35

F

failed tests 42
 failsafe-maven-plugin 266
 fallback strategies 82
 fast feedback cycles 178
 fast-running builds 225
 favorite static imports 182–183
 feature branch-driven CI 238
 feature branching
 CI 238–246
 Lone Ranger problem 243
 feature teams 78–79

FEST (Fixtures for Easy Software Testing) framework, acceptance testing with 272–283
 application 272–273
 gluing tests and processing document 276–278
 running tests 278–283
 specification 273–276

FEST tool 56

File Transfer Protocol communication. *See* FTP

fileSet element 166

filesystem 134

filtering 104

FindBugs 221, 223

Fine-grained permission system 94, 99

finger pointing 77

first-class citizens 307

FishEye browser 216

FishEye tool 56, 105–107

FishEye, integrated into JIRA 92, 97–98, 106

Fit (Framework for Integrated Test) framework, acceptance testing with 272–283
 application 272–273
 gluing tests and processing document 276–278
 running tests 278–283
 specification 273–276

Fit spec 281

Fit tool 56

fit.ActionFixture 273

Fit/FitNesse model 276

FitNesse server 283–284, 296

FitNesse tool, BDD in 283–296
 GivWenZen tool 285–290
 scenarios 293–296

FitNesseRoot folder 283

fixed constants 36

Fixtures for Easy Software Testing framework. *See* FEST

flat files 253

flight-scheduling program 293

flightArrivesAt method 291

floating labels 51

flow, and locking 90–91

fluent interface 275

folder structures 135

force attribute 90

forms, specs2 library 313–314

Fowler, Martin 43

Framework for Integrated Test framework. *See* Fit

frozen zone 79–81, 136

FTP (File Transfer Protocol) communication
 with Ant language 195–199
 with Java language 199–200

FTP client 200

FTP server 195

full release 70

full-fledged repository manager 225

fully data-driven 253

function point analyses 69

functional release management 73

functional tests 7, 46, 73, 254

G

Gaelyk framework 300

GAV coordinates 125

GDK (Groovy Development Toolkit) 301

generate-assembly profile 165, 167

generate-sources phase 281

Git 105

git blame 153

Git push 244

Git tool 41, 86

Git VCS, and git-svn bridge for feature branch-driven CI 238–246

git-svn bridge, Lone Ranger problem 243–246

GitHub 153

Gitweb 153

given component 182

Given phrase 258

given/when/then format 257

GivWenZen tool 285–290

GivWenZenExecutor 292–293

GivWenZenForSlimFixture 294

gluing, tests 276–278

GnuPG project 161–162

goals, lifecycles, phases and 128–130

Gradle 141

Gradle framework 300

Grails applications 303

Grails framework 300

GreenHopper burn-down charts 102

GreenHopper planning board 102

GreenHopper view 99, 102

Griffon framework 300

Groovy artifact 299

Groovy class 299

Groovy code 299

Groovy Development Toolkit. *See* GDK

Groovy ecosystem 305

Groovy language, Agile practices and polyglot programming 298–306
 DSLs 302–303
 Groovy Maven ecosystem 305–306
 low ceremony and scripting 301–302
 testing with easyb and Spock frameworks 303–305

Groovy meta-programming 299, 305

Groovy scripting 302

groupId coordinate 133, 142–143, 175–176

Growing OO Software 181

H

hardening phase 79

headless running mode-running tests 172

helper method 270

high-level processes 14

high-level solution 95

high-level tasks 103

highlighting 104

History object 179

history.rememberSearch(word) 184

hooks. *See also* triggers 86, 88–89

horizontal slicing 74

HTML elements 266

HTML files 253

HTTP communication protocol 144

HTTP GET requests 146

hub infrastructure 19

Hudson Trac Plugin 115

Hudson/Artifactory integration 145

Hudson/Jenkins plug-in 284

Hudson/Jenkins server 107, 110

I

IaaS (Infrastructure as a Service) 19–20

id element 140

ID generate-assembly 165

- ID nightly 269
 - IDE view 96, 104
 - idle time 90
 - IEEE (Institute of Electrical and Electronics Engineers) 64
 - IM (instant messaging) 45
 - impediment list artifact 67
 - implicit variable 227
 - imports, static 182–183
 - improvement,
 - continuous 49–50
 - inception phase 63
 - include feature 295
 - include.projects element 171
 - increments 37
 - independent release management function 122
 - individual class statements 225
 - Infrastructure as a Service. *See* IaaS
 - inheritance, and aggregation 127–128
 - init method 211
 - initMocks method 183
 - inspections 46–47
 - instant messaging. *See* IM
 - Institute of Electrical and Electronics Engineers. *See* IEEE
 - integrated toolchains 250
 - integrated tools 28
 - integration 7, 119–169
 - and release management function 120–123
 - Maven tool
 - feature set 124–132
 - Maven tool, component repositories 132–150
 - of TeamCity tool 203–205
 - integration build 31, 45, 52
 - integration management 19
 - integration test 129
 - integration tools 14
 - IntelliJ IDEA 104, 188
 - interfacing, application containers with Cargo tool 185–187
 - isEmpty method 308
 - isolated code stream 136
 - isolated sandbox 172
 - isolating systems, with Mockito tool 177–185
 - antipatterns 184–185
 - behavior-driven development 182
 - equals() method 182–183
 - favorite static imports 182–183
 - in test-driven development 180–182
 - isolation and dependency injection 178–180
 - organizing static imports 182–183
 - isolation, and dependency injection 178–180
 - items field 303
 - iterations 37, 63
 - ITIL (IT infrastructure library) 64
 - Ivy modules 141
- J**
-
- J2EE/JEE web application 134
 - JAR files 162
 - java-jar ./lib/fitnesse.jar 283
 - Java artifacts 193
 - Java class 197, 298
 - Java classpath approach 176
 - Java Content Repository. *See* JCR
 - Java language 306
 - FTP communication with 199–200
 - property editors, and GivWenZen tool 290–292
 - Java libraries 122
 - Java property files 210
 - Java syntax 304
 - Java Virtual Machine. *See* JVM
 - java.beans.PropertyEditor 290
 - java.util.List 303
 - javadoc artifacts 166
 - Javadoc documents 151
 - Javadoc files 165
 - Jazz Integration Architecture 25
 - JCL (job control language) 195–196, 198–199
 - JCR (Java Content Repository) 141
 - Jenkins Artifactory plug-in 229
 - Jenkins build job page 236–237
 - Jenkins build number 233
 - Jenkins Build Pipeline plug-in 215
 - Jenkins configuration panel 229
 - Jenkins dashboard 215–216
 - Jenkins Deploy plug-in 215
 - Jenkins integrations 222
 - Jenkins server 213–238
 - and triggering jobs 214–215
 - auditing 218–225
 - with Jenkins server 218–223
 - with Sonar tool 223–225
 - injecting build numbers into applications 227–229
 - installing and configuring Jenkins/Artifactory bridge 229–230
 - atomic deployments with Jenkins/Artifactory bridge 231–232
 - Jenkins dashboard and Jenkins jobs 215–218
 - running build fragments in 225–227
 - staging artifacts
 - and promoting 236–238
 - in Artifactory repository manager 232–236
 - Jenkins user interface 230
 - Jenkins workspace 217
 - jenkins.build_number key 227
 - Jenkins/Artifactory bridge
 - installing and configuring 229–230
 - atomic deployments with Jenkins/Artifactory bridge 231–232
 - Jenkins/Artifactory integration 236
 - JetBrains bug tracker 188
 - JetBrains TeamCity 187
 - JIRA
 - Bamboo integration in 108
 - FishEye integrated in 92, 97–98, 106
 - JIRA admin screen 100
 - JIRA connector 116
 - JIRA issues 106
 - JIRA plug-in 99
 - JIRA state/workflow system 102
 - JIRA Studio 109
 - JIRA ticket 102
 - JIRA tool 99–103
 - JIRA versions map 101
 - jiraissues macro 101
 - jiraportlet macro 101
 - job control language. *See* JCL
 - jobs
 - Jenkins server, Jenkins dashboard and 215–218
 - triggering, Jenkins server and 214–215
 - JSON object 233
 - JUnit format XML output 280

- JUnit tests 130, 150, 260
 - JUnit XML files 284
 - JVM (Java Virtual Machine) 45, 132
 - JVM ecosystem 306
- K**
-
- Kanban 65
 - Kaplan, Robert S. 22
 - key/value pairs 227
- L**
-
- labels 51
 - language-the Rhino JavaScript engine 301
 - languages, ubiquitous 255–256
 - Larman, Craig 44
 - last-minute bugs 80
 - late binding 48
 - LDAP (Lightweight Directory Access Protocol) 99
 - LDAP groups authorization 143
 - Lean approach 52
 - Lean software development 52
 - Leffingwell, Dean 78
 - legacy code 23
 - libraries 55
 - lifecycle activities 14
 - lifecycle management 123
 - lifecycles phases, goals and 128–130
 - Lightweight Directory Access Protocol. *See* LDAP
 - lightweight tooling 26–30
 - Linux-based systems 161
 - listener collaborator 184
 - Lister, Timothy 52
 - load modules 194
 - local environment 42
 - local Git repository 244–245
 - local servers 42
 - locking 86, 90–91
 - Lone Ranger problem 243–246
 - low ceremony 301
 - lower-left quadrant 252
- M**
-
- m2eclipse Eclipse plug-in 175
 - machine object 309
 - macro 101
 - magic barrel 36
 - managed environment 205
 - management support 99
 - Mantis 103
 - manual acceptance tests 83
 - Martin, Robert C. 47
 - mashup 23, 29
 - Matrix project 218
 - Maven 101
 - Maven AntRun plug-in 280
 - Maven approach 174
 - Maven artifacts 231
 - Maven Assembly plug-in 162–167
 - Maven based project 193
 - Maven build 237
 - Maven Cargo plug-in 266
 - Maven compile classpath 220
 - Maven Dependency plug-in 165, 225
 - Maven Deploy plug-in 158
 - Maven ecosystem, with Groovy language 305–306
 - Maven Enforcer plug-in 160
 - Maven Failsafe plug-in 130
 - Maven FitNesse plug-in 284
 - Maven GPG plug-in 161–162
 - Maven Jetty plug-in 213
 - Maven modules 217
 - Maven POM 228, 305
 - Maven profiles 226
 - Maven project 220, 223
 - Maven property 228
 - Maven Release plug-in 151–152, 160, 212
 - Maven SCM plug-in 152
 - Maven scripts 199
 - Maven Site plug-in 152
 - Maven site, running tests with
 - Maven tool and adding to 280–283
 - Maven Surefire plug-in 265, 281, 283
 - Maven tool 54, 56
 - component repositories 132–150
 - Artifactory repository manager 137–145
 - managing sources and binaries in conjunction 135–137
 - using subversion to serve 145–150
 - feature set 124–132
 - ecosystem 131–132
 - inheritance and aggregation 127–128
 - lifecycles, phases, and goals 128–130
 - POMs and dependencies 125–126
 - testing 130–131
 - releasing with 150–169
 - creating branch and preparing 152–157
 - final tooling 167–169
 - Maven tool plug-ins 160–167
 - testing 158–159
 - using cryptography with 161–162
 - running tests with and adding to Maven site 280–283
 - Maven-based build 232
 - maven.test.skip property 129
 - media, minimizing 50
 - meetings
 - daily Scrum 76–77
 - release planning 78
 - META-INF folders 177
 - meta-measurement 57
 - meta-model 32
 - meta-programming 301
 - metrics 74
 - Microsoft .NET 200
 - Microsoft Excel, data-driven testing with 270–272
 - Microsoft's Team Foundation Server 192
 - Mills, Harlan 37
 - minimalistic approach 111
 - minimizing media 50
 - mirrorOf element 140
 - Mock object 178
 - mocking capabilities 301
 - mocking technique 171, 177
 - Mockito features 178
 - Mockito framework 182
 - Mockito tool, isolating systems with 177–185
 - antipatterns 184–185
 - behavior-driven development 182
 - equals() method 182–183
 - favorite static imports 182–183
 - in test-driven development 180–182
 - isolation and dependency injection 178–180
 - organizing static imports 182–183
 - MockitoJUnitRunner 183

mocks 43
 modules section 167
 Money class 310
 monolithic build job 215
 moving targets 51
 MSBuild framework 201
 MSBuild system, using to build
 .NET framework projects 202–203
 Multi project 235
 multimodule project 161
 multiple check-ins 94
 Multiproject support 111
 MVC structure 300
 mvn clean install site command 282
 mvn clean jetty 213
 mvn clean package 212
 mvn deploy 143
 mvn install 130
 mvn release 157–158
 myFirstRelease branch 153
 Mylyn framework 92, 97, 103
 Mylyn monitors 103
 Mylyn subprojects 104
 Mylyn tool 56, 103–105

N

name element 147
 needs-lock property 91
 .NET artifacts 201
 .NET framework 201–207
 using MSBuild system to build projects 202–203
 using TeamCity tool to trigger builds 203–207
 build agents and cloud computing 205–207
 integration 203–205
 visual configuration environment 203
 .NET projects 205, 214
 network filesystem 150
 nightly build 31
 non-Java artifact types 193

O

object-oriented language 298, 307
 obligatory character 83
 one size fits all infrastructure 15
 one-medium approach 86–88

OnlineTranslator 179
 open Agile ALM infrastructure 193
 Open Services for Lifecycle Collaboration. *See* OSLC
 open source culture, and standards 25–26
 Open source project teams 145
 open source tools 28
 Open Systems Interconnection. *See* OSI
 OpenPGP Message Format 161
 OpenPGP standard 161
 organizational borders 12
 origArriveTime 295
 origDepartTime 295
 OSGi apps 131–132
 OSGi bundles 128, 131
 OSI (Open Systems Interconnection) 199
 OSLC (Open Services for Lifecycle Collaboration) 25, 32, 105
 Outlook tasks 115
 outside-in approach 5–6, 254
 outside-in development 21–22

P

PaaS (Platform as a Service) 19–20
 Page button, Trac entry page 113
 panta rhei 57
 peopleware 6
 personalized configuration settings 171
 phases, lifecycles, goals and 128–130
 ping-pong programming 181
 pipeline 109
 plain-process 15
 Planner tool, Agile 105
 planning board, GreenHopper 102
 Platform as a Service. *See* PaaS
 platform-opaque content 211
 platform-specific resource folder 211
 plug-ins 22–23
 Agile-Trac 111
 AntRun 199
 Archetype 177
 Artifactory 230
 Assembly 162
 Bamboo 109, 215
 Cargo Maven2 185
 Checkstyle 221–223
 Confluence 101
 Eclipse 103, 116, 131, 188
 Enforcer 161
 Failsafe 265
 FishEye 106
 Hudson/Jenkins 284
 Jenkins Artifactory 229
 Jenkins Build Pipeline 215
 Jenkins Deploy 215
 JIRA 99
 m2eclipse Eclipse 175
 Maven AntRun 280
 Maven Assembly plugin 162–167
 Maven Cargo 266
 Maven Dependency 165, 225
 Maven Deploy 158
 Maven Enforcer 160
 Maven Failsafe 130
 Maven FitNesse 284
 Maven GPG 161–162
 Maven Jetty 213
 Maven Release 151–152, 160, 212
 Maven SCM 152
 Maven Site 152
 Maven Surefire 265, 281, 283
 Maven tool 160–161
 Release 152, 156–157
 Selenium Maven 265
 versions 156
 wagon-svn 147
 polyglot programming 23–24, 298–306
 POM build section 220
 POM snippet 218
 POM version 210, 212, 237
 pom.xml files 148, 157
 POMs (project object models), and dependencies 125–126
 post-unlock hook trigger 89
 postcommit hook trigger 89
 postlock hook trigger 89
 pre-check-in tests 47
 pre-unlock hook trigger 89
 precommit hook trigger 89
 prelock hooks 91
 prepare-release profile 167
 prepareRelease parameter 167
 press command 274
 private builds 31, 42
 problematic change-sets 240
 process models 35

process pitfall 53–57
 change enablers 57
 configuration items 54–57
 effectiveness and efficiency 54
 processes 14, 54
 produced modules 233
 product backlog artifact 67
 product category 14
 product owner role 67
 production operators 18
 production-level repositories 141
 productive development environments 42
 productive workspaces 173
 programming, polyglot 23–24
 project activities 298
 project artifacts 55
 project management
 Agile 99
 Agile strategies and 36–40
 with Trac Tool 109–116
 installing Trac tool 111–112
 Roadmap tab and tickets 115–116
 timeline and sources 114–115
 wiki 113
 project managers 97
 project object models. *See* POMs
 project phases 298
 project roles 298
 project tracking tool, JIRA 56
 promoting, artifacts
 and staging 236–238
 promotion build 232
 properties build.name 145
 properties.xml file 283
 property editors, Java language
 and GivWenZen tool 290–292
 PropertyEditor 290
 proxy repository 133
 public boolean createFlight(Calendar departureTime) 291
 public Object Given(String methodString) 287
 public repository 132
 Published Modules tab 235
 pyramid of steadiness 15

Q

QA testing 18
 quality 51–53
 quality gates 47, 53, 83–84, 316
 quality pitcher 36

R

rapid application development 107
 Rational Quality Manager 25
 Rational Requirements Composer 25
 rational unified process. *See* RUP
 RCP (rich client platform) 131
 ready to go tool suites 29
 real-time notifications 97
 Redmine tool 111
 Registrations class 303
 regression tests 252
 release 68–70, 156–158, 167
 commitments 75–76
 content of 72–74
 and tests 73–74
 release container and release 0 73
 duration of 70–72
 perform goal 157
 planning meeting 78
 planning vehicles 84–86
 calendar 84–85
 screenplay 85–86
 prepare goal 156, 167
 stage goal 158
 types and categories of 70
 release backlog 75
 release category 70
 release cycles 51–53
 release durations 71
 release kickoff 66
 release management 10, 19, 73, 119–169
 integration and release management function 120–123
 Maven tool, releasing with 150–169
 release manager 18
 Release plug-in 152, 156–157
 release.properties file 158
 release/build/deploy/config block 11
 releaseProfiles element 212
 releasing, with Maven tool 150–169
 creating branch and preparing release 152–157
 cryptography 161–162
 final tooling 167–169
 Maven tool plug-ins 160–167
 testing 158–159

remote (HTTP) server 157
 remote builds, with TeamCity integration server 187–190
 remote repository 132
 repo1-cache 140
 reporting section 126, 219
 repositories synchronize people 123
 repositories, component 50–51, 132–150
 Artifactory repository manager 137–145
 managing sources and binaries in conjunction 135–137
 using subversion to serve 145–150
 repository browser 235
 repository element 147
 repository event 88
 repository manager 134
 repositoryKey 236
 reproducibility 4
 requirement-based approaches 69
 requireReleaseDeps rule 160
 resources/costs pitcher 36
 retrospectives 77
 return on investment. *See* ROI
 RIA (rich internet application) 259
 rich client platform. *See* RCP
 rich internet application. *See* RIA
 risk management 15
 Roadmap tab and tickets
 Trac tool 115–116
 ROI (return on investment) 6, 15
 rolling back approach 83
 rolling wave planning 75
 Royce, Winston W. 36
 rules, enforcing with hooks 88
 run mvn clean install 230
 runtime configuration settings 40
 RUP (rational unified process) 63

S

SaaS (Software as a Service) 19–20
 safety checks 77
 sandbox (your workspace) 93
 sandboxes 170
 sanity checks 47

- sanity test 31
- Scala artifacts 315
- Scala language
 - build ecosystem of 315
 - features 306–307
 - Forms 313–314
 - specs2 library 307–312
- Scala wrapper classes 307
- ScenarioLibrary 296
- scenarios 293–296
- SCM (software configuration management) 3, 8–13, 55
 - basics of 8–10
 - development of 10–13
- SCM checklist 56
- SCM configuration 152
- scope pitcher 36
- scopes 47
- screenplays, release planning 85–86
- scripting 301–302
- Scrum 35, 76–77
- Scrum Agile methodology 37
- Scrum framework 16
- Scrum master role 67
- Scrum release management 61–91
 - implementing 68–84
 - delivery slots, frozen zone, and code freeze 79–81
 - feature teams, component teams, and caretakers 78–79
 - progress and size of working units 74–75
 - quality gates 83–84
 - release 68–70, 75–76
 - staging software 81–83
 - synchronization
 - points 76–78
 - introduction to 62–68
 - release planning
 - vehicles 84–86
 - calendar 84–85
 - screenplay 85–86
 - Subversion version control system 86–91
 - flow and locking 90–91
 - hooks 88–89
 - one-medium
 - approach 86–88
- Scrum template 67
- SDLC (software development lifecycles) 13, 29
- secure copy (scp) 134, 159
- Selenium Core 264
- Selenium framework, testing
 - web UI with 264–269
- Selenium Maven plug-in 265
- Selenium RC (Selenium Remote Control) 264, 266
- Selenium tool 56
- self-hosted proxy repository 134
- self-hosted repository managers 145
- sell() method 309
- semantic correctness 172
- server-based VCS 242
- serverHost 267
- serverPort 267
- servers, Jenkins. *See* Jenkins server
- service orientation, SaaS, IaaS, PaaS 19–20
- service provider interface. *See* SPI
- service-oriented architecture. *See* SOA
- setAsText 290
- settings.xml file 124, 139
- SetUp wiki page 294
- shadow processes 29
- shared data problem 10
- short release cycles 71
- Siemens Corporate Research and Technologies 32
- signal files 197
- silos 4, 29
- single-select dropdown 142
- single-sourcing product information 256
- site element 135
- site lifecycle 128
- slicing of features 74
- slicing of functionality 74
- SLIM test system 284–285
- SmartDictionary 179–180
- smoke tests 31, 42–43, 47
- snapshotRepository element 135, 147
- snapshots artifacts 135
- SOA (service-oriented architecture) 20
- Software as a Service. *See* SaaS
- software configuration management. *See* SCM
- software craftsmanship 47
- software development lifecycles. *See* SDLC
- software engineering, evolution of 7–17
 - Agile 15–17
 - ALM 13–15
 - SCM 8–13
- software, staging 81–83
- Sonar tool, auditing
 - with 223–225
- Sonatype 175
- sorting 104
- source repositories 50
- Source tab 106
- source-control repository 98
- sources 51
 - managing in conjunction with binaries 135–137
 - Trac tool 114–115
 - tracking changes with Fisheye tool 105–107
- specification by example 73
- specification-based DSL 304
- specification-oriented technique 257
- specifications
 - executable 256
 - processing 273–276
- specs2 library 307–312
- SPI (service provider interface) 146
- spiral model 35
- Spock code 304
- Spock framework, testing
 - with 303–305
- Spring support, for JUnit 172
- sprint (release) backlog artifact 67
- sprints 63
- src/test/java folder 130, 260
- stability 16
- stable states 57
- staged builds 123, 215
- staging artifacts
 - and promoting 236–238
 - in Artifactory repository manager 232–236
- staging build 232
- staging in Artifactory 232
- staging ladder 82
- staging software 81–83
- stagingRepository parameter 158

- stakeholders
 - focus on 17–19
 - using Subversion 87
 - standard distribution, Eclipse 104
 - standard TestNG report 280
 - standards 25–26, 51, 53
 - start command 273
 - Start-commit hook trigger 89
 - state object 294
 - state-of-the-art releasing 232
 - state/workflow system, JIRA 102
 - static HTML 101
 - static imports 182–183
 - status accounting 55
 - status accounting function 9
 - steadiness pyramid 15–16
 - stealing locks 90
 - story-based DSL 304
 - strategies 6, 35
 - streams, coding single 40–41
 - structural database changes 85
 - structural releases 70
 - submodules 160
 - suboptimal collaboration 12
 - Subversion hooks 137
 - Subversion repository 87, 156
 - Subversion revisions 115
 - Subversion tool 41
 - Subversion VCSs 146, 239
 - Subversion version control system 86–91
 - flow and locking 90–91
 - hooks 88–89
 - one-medium approach 86–88
 - Subversion-based Maven repository 148
 - subversion, using to serve repository 145–150
 - Sun/Oracle code conventions 222
 - Super POM 127, 140
 - svn 147
 - svn copy 152
 - SVN hooks 156
 - SVN repository 151
 - SVN tag location 154
 - svn unlock word.doc 90
 - Swing application 272
 - SWT applications 250
 - synchronization 49–50
 - synchronization points 76–78, 123
 - daily Scrum 76–77
 - release planning meeting 78
 - retrospective 77
 - system infrastructure 95
- T**
-
- TableRowSorter 272
 - TAG folder 158
 - tagBase element 154
 - .tar.gz format 163
 - task AGILEALM-10 108
 - task view, Eclipse 103
 - task-based build process 31
 - task-based development 4–5, 21–22, 92–116
 - prerequisites for 93–97
 - changesets 94–95
 - coordinating changes 93–94
 - toolchains 97–109
 - build view with Bamboo server 107–109
 - Eclipse and Mylyn tools 103–105
 - JIRA tool 99–103
 - tracking source changes, with FishEye tool 105–107
 - Trac tool, bug tracking and project management 109–116
 - installing 111–112
 - Roadmap tab and tickets 115–116
 - timeline and sources 114–115
 - wiki 113
 - tasks 21
 - associating changesets with 95–97
 - scheduling 104
 - TDD (test-driven development) 180, 185, 252
 - TDD practices 303
 - TDD style 257, 285
 - Team Foundation Server 205
 - TeamCity 201, 203–205
 - TeamCity dashboard 189–190
 - TeamCity integration server, remote builds with 187–190
 - TeamCity tool, using to trigger .NET framework builds 203–207
 - teamicide techniques 52
 - teams
 - Agile 99
 - component 78–79
 - feature 78–79
 - role of 67
 - technical implementation view 138
 - technical release management 73
 - technical releasing 86
 - technical tests 46
 - technology-facing tests 251
 - TELL interactions 184
 - temporary files 157
 - test matrix 251
 - Test spy 178
 - Test stub 178
 - test-driven development. *See* TDD
 - testAdd() method 258
 - testDataTable method 268
 - testing 249–296
 - acceptance, with Fit, TestNG, and FEST frameworks 272–283
 - automatic 46–47
 - BDD in FitNesse tool 283–296
 - GivWenZen tool 285–290
 - scenarios 293–296
 - collaborative 251–259
 - acceptance tests 254–255
 - BDD 256–259
 - data-driven tests 253–254
 - executable specifications 256
 - ubiquitous language 255–256
 - data-driven 259–272
 - testing web UI 264–269
 - TestNG framework and data-driven approach 259–262
 - with Microsoft Excel 270–272
 - with XStream library 262–264
 - Java code 298
 - with easyb and Spock frameworks 303–305
 - with Maven tool 130–131, 158–159
 - TestNG framework
 - acceptance testing with 272–283
 - application 272–273

- gluing tests and processing
 - document 276–278
 - running tests 278–283
 - specification 273–276
- and data-driven
 - approach 259–262
 - testing web UI with 264–269
- TestNG test method 270
- TestNG test tool 259–260, 265–266, 283
- testng-firefox-minimal.xml 265
- testng.xml files 260
- TestNG's DataProvider 261
- tests 12
 - content of release and 73–74
 - gluing 276–278
- text-based JIRA approach 102
- then steps 290
- ticketing system 96
- tickets, Trac tool 115–116
- time pitcher 36
- time-based estimations 69
- timeboxed approach 68
- timed builds 80
- Timeline tab 114
- timeline view 114
- timeline, Trac tool 114–115
- tool integrations 95
- tool-centric approach 15
- toolchains 4–5, 12, 26, 28–29, 97–109
 - build view with Bamboo server 107–109
 - Eclipse and Mylyn tools 103–105
 - JIRA tool 99–103
 - tracking source changes with FishEye tool 105–107
- tooling 54
 - final before release 167–169
 - lightweight 26–30
- tools 22
- TortoiseSVN 152–153
- toString() method 310
- toUpperCase method 302
- Trac Browse Source tab 115
- Trac project 111
- Trac roadmap 116
- Trac timeline 115
- Trac tool, bug tracking and project management
 - with 109–116
 - installing Trac tool 111–112
- Roadmap tab and tickets 115–116

- timeline and sources 114–115
- wiki 113
- Trac toolchain 92, 103, 109–111, 116
- Trac wiki 113
- traceability 4, 6, 14, 96
- transition phase 63
- Translator object 179
- translator.translate(word) 184
- transparency 17
- transparent processes 12
- triggers 86
- tringTokenizer 271
- triple double-quotes 314
- turnaround times, reducing 46

U

- ubiquitous language 255–256
- UI (user interface) 253
- UI controls 253
- UI tests 253
- UML diagrams 312
- unambiguous language 256
- uniqueVersion configuration
 - option 147
- unit specification style 308
- unit tests 42
- Unread tickets 103
- Unscheduled area 102
- update (pull changes) 96
- update problem 10
- url element 147
- user activity 105
- user interface behavior 272
- user interface. *See* UI
- user-defined parameters 218

V

- value pairs, Agile Manifesto 39
- value system 36
- values 6
- VCS (version-control systems) 8, 23, 64
- VCS base URL 236
- VCS branches 136
- VCS browser 98
- VCS changesets 104
- VCS head 135
- VCS in Eclipse 105
- VCS mainline 241
- VCS setup 244
- VCS system 93–96, 98, 109

- VCS tag 136
- VCS tag base URL 236
- VCS tools 86
- vehicles, release planning 84–86
 - calendar 84–85
 - screenplay 85–86
- verbose build comparison 107
- verifyNoMoreInteractions()
 - method 184
- <version/> XML element 153
- version control 40–41, 55
- version coordinate 142–143, 175
- version element 126
- version history features, Subversion 87
- version management servers 51
- version-control systems. *See* VCS
- versioned artifacts 135
- versions
 - lock-snapshots plug-in 156
 - unlock-snapshots plug-in 156
- versions map, JIRA 101
- vertical slicing 74
- virtual machine images 112
- virtual repositories 138
- virtuous cycle 48
- visual configuration environments 203
- VMWare 20
- Vodde, Bas 44

W

- Wagon mechanism 147
- wagon-svn plug-in 147
- waiting times 90, 208
- WAR files 209, 212
- waterfall model 35–36
- WBS (work breakdown structure) 75
- web application administrator 299
- web apps, for multiple environments 207–213
 - applying different configurations 210–212
 - multiple artifacts with assemblies 208–210
 - using distribution profile and executing example 212–213
- web UI (user interface), testing 264–269
- web-based reporting 105

WebDAV features 134
WebSVN 152
Weinberg, Gerald M. 52
white box tests 252
wiki
 Atlassian Confluence 101,
 109–110
 Trac tool 113
Word documents 256
work breakdown structure. *See*
 WBS
work items 21
working in isolation 42

working offline 104
working units, progress and size
 of 74–75
workspace management, con-
 gruent builds and 171–177
workspaces, productive 42–43
write access rights 159

X

XML data 262
XML document 279
XML metadata 142

XML structure 262–263
xmlfileset 278
XStream library
 data-driven testing
 with 262–264
 testing web UI with 264–269

Z

zero tolerance approach 53
.zip format 163
zipvalidate function 304