Index

Numerics
4 + 1 View Model of Architecture, The (Kruchten), 88

A
Acceptance tests, 342, 349
Active code generators
  benefits, example of, 121
  boilerplate code, J2EE projects, 121
  capabilities, 119
  defined, 114
  elements of, 119
  metadata, 119
  pattern template, 119
  project agility, 121
  rapid development, as tool for, 121
  success factors, 120
Activity diagrams, 92
Actors, 38, 91
Ada, 195
Adaptive foundation
  benefits, 10
  best practices and techniques, 3, 27, 46
  change, impact of, 7
  defined, 6
  investment concerns, 7
  methodologies, as definitive key, 28. See also Methodologies
    rapid development and, 3, 4
    requirements, 3, 6
Adaptive methods, 29. See also Iterative development
Advice, 236
Aggregation, 94
Agile Database Techniques (Ambler), 141
Agile DBA, 141
Agile methods, 47. See also XP (Extreme Programming)
Agile Modeling process, 111
Agile movement, 17, 27, 289
Agility, 212
Ambler, Scott, 111, 141
AndroMDA
  Ant build files, 180, 185
  cartridges, pluggable MDA, 180, 184
  code generation, 179
  deployment descriptors, 186
  development of, 178
  download site, 178
  handwritten code, safeguarding, 190
  MDA process with, 179
  Poseidon plug-in, 181
  prototyping with, 190
reverse engineering, PSM to PIM, 190
stereotypes, 182
templates, 180, 186, 188
test cases, generating, 329
Velocity and XDoclet, use with, 179, 186
version 3.0, 191
VTL scripting objects, 188
XML Metadata Interchange (XMI) format, 180, 181
Annotations, 123
Ant tasks, 265
Antgraph, 273
Apache, 66
Apache Ant. See also Build process
Ant tasks, 265
Antgraph, 273
build dependencies, 267
conditional build dependencies, 269
crossplatform support, 265
delegate-out versus include-type functionality, 271
delegating build file, 278
IDE integration, 281, 301
Jython, extending with, 282
key features, 266
project directory structure, 276
subprojects, working with, 271
target-naming conventions, 274
XML-based syntax, use of, 265
Apache Axis, 79
Apache Struts, 6, 174
Apache Tomcat, 208, 312
Apache Velocity
AndroMDA, use with, 179, 186, 188
code wizard, example, 115
data mapping, with code generator, 116
directives, VTL, 116
Middlegen, 152
output generated, 118
references, VTL, 116
template, defining, 115
uses, 115
Application layer, 70
Architect
ideal, 62
job of, 61
Architectural significance, 91
Architecture
composition and aggregation, 94
defined, 61
delivery timeframes, 63
distributed, 75. See also Distributed computing
multiple views of, 88
multitier, 70, 73
overengineering solutions, 63, 69
prototyping, 82
RAD, designing for, 61. See also RAD, architecture and design
software reuse, 62
system traits, 62
UML diagrams, 88, 89
versus design, 62
view types, 89
Architecture. See also EJB architecture
ArcStyler, 178
ArgoUML, 181
AspectJ
AOP paradigm, first implementation of, 238, 244
compiler, 239
Eclipse and, 238, 239, 243
example, 239
language-based approach, 231, 238
plug-ins available, 239
pointcut designators, 241
weaving methods supported, 239
AspectJ Development Tool (AJDT), 238, 243
Aspect-oriented middleware, 231, 251
Aspect-oriented programming (AOP)
adoption strategies, 251
advice, 236
aspects, 236
aspects and classes, relationship between, 236
benefits, 231
crosscutting concerns, modularizing, 235
framework implementations, listed, 245
hunchbacks and dragons allegory, 237
join points, 236
language versus framework, 245
MDA and, 253
pointcut designators, 241
pointcuts, 236
weaving, 236
weaving methods, 237
Aspect-oriented programming (AOP). See also AspectJ. See also AspectWerkz. See also Crosscutting concerns
Aspect-Oriented Programming: A Critical Analysis of a New Programming Paradigm (Highley), 236
Aspects, 236
AspectWerkz
AspectJ, compared, 246, 249
aspect-oriented middleware, 251
framework-based approach, 231
Metadata annotations, 249
metadata annotations, as aspects, 246, 249
offline and online modes, 250
pointcut designators, 248
weaving options, 250
XML aspect definition files, 246, 248
Attribute-oriented programming
.NET platform support, 123
attributes, defined, 122
code, annotation with attributes, 123
J2SE 5.0 annotations, 123
JSR-175, A Metadata Facility for the Java Programming Language, 123
JSR-181, Web Services Metadata for the Java Platform, 123
metadata facility example, 123
Microsoft platform support, 123
preprocessor directives, 123
runtime attributes, 123
XDoclet. See XDoclet
Automated testing. See also HttpUnit API
acceptance tests, 349
costs, 346
functional tests, 349
J2EE challenges, 347
open source tools, listed, 348
reusability, of test scripts, 346
tool categories, 347
Automatic deployment, 264
Avitek Medical Records (MedRec) example, 351

B
Backward chaining system, 220
BEA WebLogic Server, 129
Bean-managed persistence (BMP), 148
Beck, Kent, 48, 54, 56, 58, 320, 339
Behavioral prototypes, 23
Big-bang approach, 18
Bill, Robert W., 210
Binder, Robert, 345, 369
Black-box testing, 332, 342, 350
Blaze Advisor, 225, 226, 228
BMP (bean-managed persistence), 148
Boehm, Barry, 33
Bohlen, Matthias, 178, 191
Boilerplate code, 121, 125
Bonér, Jonas, 246
Booch, Grady, 86, 109
Borland’s Together ControlCenter, 134, 154
Brooks, Frederick, 169
Build process. See also Apache Ant
  automatic deployment, 264
  builds and deployments, minimal, 263
  common activities, 261
  continuous software integration, 285
  declarative approach, 266
  designing, main requirements, 261
  efficiency-focused mindset, 259
  hot deployment, 264
  incremental tasks, 263
  J2EE build tasks, 262
  packaging, J2EE components, 263
  process improvements, 260
  project organization, 276
  recurring tasks, 260
  reusability, 261
  virus checkers, effect of, 269
Burke, Eric, 273
Business Delegate pattern, 104
Business Rule Management System (BRMS), 225
Business rules. See also Rule engines
  collective purpose of, 211
  defined, 212
  dynamic nature, 212
  enterprise systems support, need for, 213
  examples, 212
  structure, 212
  traditional systems, limitations, 213

C
  C3 project, 48, 52, 56
  Call-time weaving, 237
  Cargo cult software development, 99
  Case Against XP, The (Stephens), 53
  CASE tools. See also Modeling tools
    emergence of, 85
    RAD, use with, 21, 85
Christmas puppy syndrome, 107
Class diagrams, 93
  Classic waterfall model. See Waterfall lifecycle model
Clean operations, 128
Client tier, 70
Client/server model, 72
CLIPS (C Language Integrated Production System), 216, 217, 220, 229
CMP (container-managed persistence), 148
Cockburn, Alistair, 58
CocoBase, 149
Codd, E. F., 143
Code Complete (McConnell), 313
Code generation. See Data access code generation (example)
  capabilities, 113
  defined, 114
  guidelines and key points, 135
  IDE code wizards, 114, 297
  JAD, 114
  Java support, 178
  modeling tools versus MDA, 172
  unit tests, 325
Code Generation Network, 138
Code injection, 237
Code model, 172
Code tangling and scattering, 232
Collaboration diagrams, 95
Collaborative groupware, 13
Comfort zones, 14
Compile-time weaving, 237
Component diagrams, 98
Component libraries, 20
Component-based architectures, 20
Composition, 94
Concern, defined 232. See also Crosscutting concerns
Container-managed persistence, 148
CORBA, 72, 170
Core J2EE Patterns (Deepak), 104
Craig, Philip, 332
CRC (Class, Responsibilities, and Collaboration) cards, 52
Critical success factors. See Rapid development, critical success factors
Crosscutting concerns
business-level, 232, 234
code tangling and scattering, 232
concern, defined, 232
defined, 232
domain-specific languages, 235
frameworks, J2EE platform, 233
mix-in classes, 234
modularization of, as AOP goal, 235
separation, of concerns, 232
system-level concerns, 232
Visitor pattern, 234
CruiseControl, 285
Crystal, 47, 58
Cunningham, Ward, 48
CVS, 289
Czarnecki, Krzysztof, 138

Data Access Object pattern, 174
Data access technologies
entity beans, 147
Java Data Objects (JDO), 149
Java Database Connectivity (JDBC), 145
Object/relational (O/R) mapping tools, 146, 150
SQL, 145
Data definition language (DDL), 154
Data layer, 70
Database problems
Agile DBA, role of, 141
application and data teams, distinction between, 140
data models driving the object model, 144
database types to Java types, mapping, 143
enterprise data, value, 140
legacy data structures, 141
object-relational impedance mismatch, 142
persistence transparency, inability to achieve, 143
relationships, mapping, 143
security and confidentiality, 142
sensitive data, inability to work with, 142
shared access between systems, 141
DBArtisan, 306
Declarative programming languages, 214, 215, 266
Delivery timeframes, 3, 7, 18, 61, 170
Department of Defense, 195
Deployment diagrams, 98
Deployment view, 89
Design patterns
Business Delegate, 104
code wizards, 289
Data Access Object 174
Session Façade 77, 174
Singleton, 233
software reuse, 104
templates, 104
Transfer Object, 77
Values Object, 174
Visitor, 234
Development environment, 343
Distributed computing
complexity, 76
J2EE architectures, 71
marshaling, 75
object-oriented impedance, 77
performance, 75
remote method calls, 76
skeleton object, 76
stub object, 75
Domain-specific languages, 19, 235
Dot layout engine, 274

E
EasyMock, 334
Eclipse platform
AspectJ and, 238, 239, 243
collection parameters, setting, 292
debugging, J2EE applications, 310. See also
Java Platform Debugger Architecture
(JPDA)
development of, 291
deployment features, 299
deployment, 293
explained, 291
hot swapping, 312
installing and running, 291
Java development tools (JDT), 238
JDT (Java Development Tooling), 291
JSP debugging, 313
JUnit tests, running, 323
perspectives, 293
plug-in tools, 294
projects, 292
unit tests, generating, 325
views, 293
workbench paradigm, 293
workspace, 292
EclipseUML, 306
EIS (Enterprise Information System) tier,
71, 79
Eisenecker, Ulrich, 138
EJB architecture
benefits, 74, 81
client views, remote and local, 75
terprise beans, 74
local interfaces, 80
location transparency, 74, 75
weaknesses, 82
EJB container, 80, 121, 148
container-managed persistence
(CMP), 148
EJB tier, 71
EJB wizard, 299
EJBDoclet, 124
Emacs, 239
Endo-Testing: Unit Testing with Mock
Objects (Mackinnon, Freeman, and Craig), 332
Enterprise JavaBeans (EJBs)
component-based architectures, 74
defined, 74
EJB 3.0 specification, 176
framework-dependent code, 121
local interface, 75
multitier architectures, 72
program artifacts, 121
remote interface, 75
specialist skills, need for, 54
versus POJOs, 79
XDoclet, development with, 125
Enterprise Web applications
benefits, 79
EIS tier, 79
limitations, 79
POJOs versus enterprise beans, 79
Web container, 78
Web-centric architecture, 77
Enterprise-level systems. See also Rapid development
aspect-oriented middleware, 231
big-bang approach, 18
building, using J2EE, 3
challenges and concerns, 4
complexity and changing nature of, 3
defined, 4
delivery timeframes, 3, 7, 18
dynamic behavior, modeling, 92
goals, 4
RUP development support, 46
Entity beans, 74, 147
bean-managed persistence (BMP), 148
Entity-relationship (ER) diagrams, 144, 153
ERwin, 306
Evolutionary prototypes, 23, 244
Executable UML, 108
Expert One-on-One J2EE Design and Programming (Johnson), 75
Expert systems, 214
Exploratory prototypes, 23
Extreme Programming Explained (Beck), 56, 58. See XP (Extreme Programming)

F
Facts and Fallacies of Software Engineering (Glass), 25
Fat client applications, 72
Feature-driven development (FDD), 47
Feynman, Richard, 100
Fix-and-continue debugging, 312
Flow of events, 38, 39, 92
Foemmel, Matthew, 285
Forgy, Charles L., 229
Forward chaining systems, 220, 227
Forward engineering, 102, 114
Fowler, Martin, 58, 83, 111, 285
Frameworks
boilerplate code, 121
maturity and longevity, assessing, 67
open source, listed, 64
rapid development, use in, 8
selection guidelines, 66
Freeman, Steve, 332
Friedman-Hill, Earnest J., 217
Functional tests, 342, 349

G
Gamma, Erich, 320
Generative programming, 138
Generative Programming: Methods, Tools, and Applications (Czarnecki and Eisenecker), 138
Gentleware, 181
Glass, Robert L., 25
GNU Public License (GPL), 153
Gosling, James, 123
Graphviz, 274
Groovy, 64, 198, 209
Guillemets, 182

H
Heuristic, 215
Hibernate, 151, 160, 174, 182
Hot deployment, 264
Hot swapping, 312
How to Fail with the Rational Unified Process: Seven Steps to Pain and Suffering (Kruchten), 42
HttpUnit API
classes, 350
JMeter, using with, 368
JUnit, using with, 351
Swing and AWT-base fat clients, 354
test writing example (MedRec), 351
Hunchbacks and dragons allegory, 237
Hunt, Andrew, 69
IBM Rational Unified Process (RUP) activities, 44
adaptive method, 37
artifacts, 44
best practices, 37
disadvantages, 47
disciplines, 43
elements, 44
enterprise projects, supporting, 46
framework versus methodology, 37
iteration plans, 45
iterative development, 41
key points, 47
phase plan, 45
phases, 41
prototyping, use of, 22
roles, 44
test-driven development, 318
timeboxed iterations, 41, 46
UML, 37, 40
UML, association with, 109
use cases. See Use cases
workflow detail, 45
workflows, 45
IDE, for J2EE projects. See also Integrated development environments (IDEs)
additional tool sets, 296
Ant integration, 301
application deployment, 305
code assist or code completion, 300
code generators, 302
code wizards, 297
code-level artifacts, 297
database access, 306
debugging, 310
debugging. See also Java Platform Debugger Architecture (JOPDA)
EJB wizard, 299
JDT, features supported, 296
modeling support, 306
multiple file types, editor support for, 299
project-level wizards, 297
server control, 305, 306
Velocity templates, 302
XDoclet, 303
Impedance mismatch. See Object-relational impedance mismatch
Imperative computational model, 214, 215
Implementation view, 88
Inference engines, 215
Integrated development environments (IDE) 114. See also Eclipse platform. See also IDE, for J2EE projects
build constraints and project structure, 293
code wizards, 289
deployment, 289
help access, integrated, 289
key features, listed, 290
language-aware editors, 288
refactoring support, 289
source control integration, 289
test code generators, including, 327
testing, 290
toolset integration, 287
versus best-of-breed approach, 288
Integration tests, 342
IntelliJ IDEA, 316
Interaction diagrams, 87, 94, 102
Interception weaving, 237
Iterations, 33, 45, 50
Iterative development. See also IBM Rational Unified Process (RUP). See also Extreme Programming
benefits, 34
explained, 33
lifecycle, 33
software reuse, 35
spiral model, 33
testing process, 344

J

J2EE (Java 2 Enterprise Edition) platform.  
See also Enterprise-level systems
business software, changing nature of, 3
component reuse, 20
crosscutting concerns, system-level, 233
domain-specific languages, use with, 19
tester solutions, building, 3, 4, 7
entity beans, as persistence layer, 148
frameworks, 64, 121
multitier architectures, 71, 216
orthogonal systems design, 71
scripting languages, 197
services and technologies, 6
specialist skills, need for, 54
testing challenges, 347
XP, using for development projects, 56

J2EE architectures
client tier, 70
distributed computing. See Distributed computing
EIS tier, 71, 79
EJB-centric architecture, 80. See also EJB Architecture
middle/business tier, 71, 73
Web-centric architecture, 77. See also Enterprise Web applications

J2EE Connector Architecture (JCA), 54, 71
J2EE design patterns (Sun Microsystems), 77

J2EE Framework, 3

J2SE (Java 2 Standard Edition), 6, 20, 123, 199
Jacobson, Ivar, 37, 40, 86, 109
Java BluePrints program, 83
Java Community Process, 123, 176, 224
Java Data Objects

J2EE Connector Architecture, 150
Java Data Objects (JDO), 149
Java Database Connectivity (JDBC), 145
Java decompiler (JAD), 114
Java development tools (JDT), 238, 291
Java Messaging Service (JMS), 71, 74
Java Naming and Directory Interface (JNDI), 71
Java Platform Debugger Architecture (JPDA)
debugging guidelines, 313
Java Debug Wire Protocol (JDWP), 308
Java Debugging Interface (JDI), 307
Java Virtual Machine Debug Interface (JVMDI), 307
JVM debug configuration, 309
JVM, attachment to, 307
remote debugging, 308
Java Remote Method Invocation (RMI), 75
Java Rule-Engine API, 224
Java ServerFaces, 174
Javadoc output, 125
Javadoc tags, 124
JBoss, 178, 305
JBoss AOP, 251
JBBuilder, 239, 281, 316
JDBC API, 71, 306
JDO. See Java Data Objects
Jefferies, Ron, 48
Jess (Java Expert System Shell)
asserting facts, 219
backward chaining, 220
CLIPS, as Java implementation of, 216
deployment, 217
forward chaining, 220
history, 216
installing, 217
Java API, 221
left-hand side (LHS) term, 219, 220
Rete algorithm, 220
Rete class, instantiating, 223
right-hand side (RHS) term, 219, 220
rules, 219
scripting language capabilities, 217
JFCUnit, 354
JMeter
aggregate reports, 365
cookie manager, 360
elements, 358
executing test plans, 365
graph display reports, 366
HTTP request defaults element, 359
HttpUnit, using with, 368
listeners, 364
load testing guidelines, 367
logic controllers, listed, 360
login requests, 362
loop controller, 361
overview, 356
results, analyzing, 365
samplers, 361
search requests, 363
simple controller, 361
test plan example (MedRec), 356
thread groups, creating, 358
jMock, 334
Johnson, Rod, 75, 83
Join points, 236
JRockit JVM, 251
JRuby, 198
JRules, 224–228
JSP, 19, 54, 289, 300, 313
JSR-012, Java Data Objects, 149
JSR-045, Debugging Support for Other Languages, 313
JSR-175, A Metadata Facility for the Java Programming Language, 123
JSR-181, Web Services Metadata for the Java Platform, 123
JVM (Java Virtual Machine), 197
Jython, 64, 217
compiling, 283
extending Ant with, 282
features, 198
glue language, using as, 206
inheritance, 204
installing, 200
interactive mode, starting in, 200
Java, integration with, 203
JVM, ability to run under, 198
prompts, primary and secondary, 201
prototyping with, 205
Python features incorporated, 199
RAD, use with, 19
script files, running, 200
servlets, 207
syntax basics, 201
Jython Essentials (Pedroni and Rappin), 210
Jython for Java Programmers (Bill), 210
K
Kiczales, Gregor, 232, 238
Kissinger, Henry, 12
Knowledge bases, 214, 215
Kruchten, Philippe, 42, 45, 46, 58, 88
L
Laddad, Ramnivas, 255
Layers, 70
Learn-as-you-go-approach, 7
Load tests, 342
Load-time weaving, 237
Location transparency, 74, 75
Logical view, 88
Lomboz, 296, 316

M
Mackinnon, Tim, 332
MagicDraw, 181
Make, 285
Marshaling, 75
Maven, 285
Mazzocchi, Stefano, 356
McConnell, Stephen, 100, 313
McWhirter, Bob, 209
Message-driven beans, 74
Methodologies
adaptive, 29. See also Iterative development
agile, 47. See also XP (Extreme Programming)
minimalist approach, 28
predictive, 29. See also Waterfall lifecycle model
RAD, 28
reasons to use, 27
Microsoft platform, 123
Middle/business tier, 71, 73
Middlegen, 152, 158, 184
Mix-in classes, 234
MockCreator, 333
MockMaker, 333
Mocks
choosing, static versus dynamic, 337
defined, 332
dynamic, example, 334
endo-testing, 332
information Web site, 339
static, 333
test stubs, compared, 332
white-box testing, 332
writing, 333
Model_Driven Architecture (MDA)
test-driven development, support of, 328
Model-centric development practices, 100
Model-Driven Architecture (MDA)
AOP and, 253
benefits, 170
business-centric approach, 169
code model, 172
compliant tools, 178. See also AndroMDA
defined, 169
goals, 170
implementation, 172
legacy systems, 177
mapping, 173, 177
marks, 180, 181
model lifecycle, 173
model-centric approach, 169, 171
PIMs (platform-independent models), 171–173
platform neutrality, 108, 170, 171
PSMs (platform-specific models), 172, 173
refinement, 172
reverse engineering, PSM to PIM, 177
strengths, 175
transformation process, PIM to PSM, 174
versus modeling tools, 172
weaknesses, 177
Modeling
advantages, listed, 99
Agile Modeling process, 111
cargo cult software development, 99
failings, 99
success guidelines, 109
Modeling tools
   choosing, 100
   Christmas puppy syndrome, 107
design pattern support, 104
executable UML, 108
features, listed, 101
forward and reverse engineering, 102, 167
listed, 101
misconceptions, 107
process-agnostic, 109
roundtrip support, 103
synchronizing model with code, 104
templates, 104
UML support, 102
validation, of interaction diagrams, 102
value of using, 100
Modeling tools, forward and reverse
ingineering, 114
Models. See also Model-Driven Architecture
(MDA). See also UML diagrams
communication mechanisms of, 86
defined, 85
platform neutrality, 108
prototyping, 88
synchronizing, with source code, 104
system architecture, aspects represented, 86
UML, as standard notation, 85
validation, of design, 87, 96
Multitier architectures, 70, 73
MyEclipse, 296, 297, 300, 303, 305
MySQL, 153

N
Neato layout engine, 274
NetBeans, 239, 316
Normalized data, 142

O
Öberg, Rickard, 124
Object containment, 94
Object database management systems
(ODBMS), 143
Object Management Group (OMG), 102,
170, 172, 178, 181, 192
Object/relational (O/R) mapping tools,
146, 150
Object-oriented impedance, 77
Object-oriented programming (OOP),
20, 232
Objectory process, 37, 40
Object-relational impedance mismatch, 142
Offshore development, 46, 56, 58, 87
OPS5 language, 217, 226
OptimalJ, 178, 179, 181, 191
Orthogonal components, 69
Orthogonal design, 69, 71

P
Package diagrams, 94
Pair programming, 52
Paper Prototyping: The Fast and Easy Way to
Design and Refine User Interfaces
(Snyder), 25
Paper-based prototypes, 25
PARC. See Xerox Palo Alto Research Center
Passive code generators
advantages, 114
Apache Velocity. See Apache Velocity
benefits of, 118
defined, 114
IDE code wizards, 114, 297
platforms migration, 118
template driven, 114
Pattern templates, 104
Patterns of Enterprise Application
Architecture (Fowler), 83
Pedroni, Samuele, 210
Performance tests, 342
Perl, 24, 64, 196, 203
PIM marks, 181
PIMs (platform-independent models), 171, 173
Pointcut designators, 241, 248
Pointcuts, 236
POJOs (Plain Old Java Objects, 79, 150
Poseidon, 181
Pragmatic Programmer, The (Hunt and Thomas), 69
Predictive methods, 29
Preproduction environment, 343
Presentation layer, 70
Primary use cases, 91, 96
Process view, 88
Production environment, 344
Production systems, 214
Prolog, 19, 216, 221
Prototypes
AndroMDA, building with, 190
approaches, choosing, 24
AspectJ, building with, 243
behavioral, 23
benefits, 22
evolutionary, 23, 244
exploratory, 23
IBM Rational Unified Process (RUP), 22
Jython servlets, 207
models and, 88
paper-based, 25
reasons to use, 21
selection factors, 22
structural, 24
throwaway, 23
timebox development, 23
PSMs (platform-specific models), 172, 173
Python, 24, 196–198, 203, 209, 217

Q
Quality assurance (QA). See also HttpUnit API
acceptance tests, 342, 349
defined, 341
functional tests, 342, 349
integration tests, 342
load tests, 342, 355
performance concerns, 355
performance tests, 342
project environment, 343
regression tests, 342
stress tests, 342, 355
system tests, 342
test-driven approach, 345
testing process, iterative development, 344
unit tests, 341
Quantum DB, 306

R
RAD (rapid application development)
component-based architectures, 20, 74
domain-specific languages, 19
emergence of, 17
methodology, ideal, 28
object-oriented programming, 20
productivity tools, 21
rapid prototyping, 21. See also Prototypes
software reuse. See Software reuse
terminology changes, rapidity to agility, 17
timebox development, 18, 23
RAD, architecture and design
frameworks. See Frameworks
goals, 63
layered approach, 70
multitier architecture, 70
orthogonal design, 69
software reuse, designing for, 68
team strengths, 64
testing, 67
type checking and XML, 68
Rapid development
adaptive foundation, as basis, 3, 4. See also
Adaptive foundation
continuous improvement, 10
frameworks, 8
investment concerns, 7
need for, 4
people-centric process, 8, 12, 47
practices, 8
processes and procedures, 9
standards, 9
techniques, listed, 288
tools, 8
training requirements, 9. See also Training
Rapid development, critical success factors
collaborative groupware, 13
comfort zones, stepping outside of, 13
developer acceptance, 12
education, 13
listed, 12
management support, 14
seminars, 13
Rappin, Noel, 210
Rational Rose, 174
Rational Unified Process, The: An
Introduction (Kruchten), 58
Refactoring patterns, 289
Refactoring tools, 87
Refinement, 172
Reflection, 321
Regression tests, 342
Relational databases, 143
Remote debugging, 289, 308
Runtime weaving, 237
Rumbaugh, James, 86, 109
RMI. See Java Remote Method Invocation
Roundtrip process, 103
Ruby, 196, 198, 203, 209
Rule engines. See also Business rules
benefits of using, 215
declarative programming languages, 214, 215
decoupling business rules, 226
developer acceptance, 12
end-user programming, 226
enterprise systems, using in, 216, 225
evaluation criteria, 227
expert systems, 214
facts, as data, 215
heuristic, 215
history, 214
imperative computational model, 214, 215
inference engines, 215
Java rule-engine API, 224
JSR-94, 224
knowledge bases, 214, 215
open source Java, listed, 229
production systems, 214
rule-based systems, 215
visual development tools, 227
S
Scenarios, 39
Scripting languages
choosing, points listed, 197
crossplatform support, 197
education and training, need for, 197
examples of, 196
features, listed, 196
JVM (Java Virtual Machine), listed, 197
standardization, 197
team expertise, 197
typing, 195
uses and tasks, 196
Scripting languages. See also Jython
SCRUM, 47
Sequence diagrams, 95  
Session beans, 74  
Session Façade pattern, 77, 174  
Singleton pattern, 233  
Skelton object, 76  
Smalltalk, 209  
Snyder, Carolyn, 25  
Software architecture document (SAD), 44  
Software reuse  
  AOP and, 235  
  architecture, as basis, 62  
  benefits, 19  
  design aims, 19  
  design patterns, using, 104  
  designing for, 68  
  iterative development, 35  
  object-oriented design, 20  
  Spiral model, 33  
  Split-site development, 56, 58  
  SQL, 19, 145  
  Staging area, 343  
Statechart diagrams, 97  
Stephens, Matt, 53  
Stereotypes, 182  
Strachan, James, 209  
Stress tests, 342  
Structural prototype, 24  
Stub object, 75  
Studio Creator, 21  
Study groups, 10  
Syntax coloring, 288  
System tests, 342  
System-level concerns, 232  

Test-driven development. See also JUnit  
  automated unit testing, 339  
  benefits, 318  
  change, impact of, 317  
  defined, 317  
  IBM Rational Unified Process (RUP), 318  
  implementation factors, 319  
  MDA support, 328  
  object isolation, 331. See also Mocks  
  quality assurance and, 345  
  unit test bloat, 319  
  XP approach, 318  
Test-first development, XP, 54  
Test-first development, XP approach, 318.  
  See also Test-driven development  
Testing environment, 343  
Testing Object-Oriented Systems: Models, Patterns, and Tools (Binder), 345, 369  
Thomas, David, 69  
Three amigos, 86  
Throwaway prototypes, 23  
Tiers  
  business, 78  
  client, 70  
  defined, 70  
  EIS, 71, 79  
  EJB, 71  
  middle/business, 71, 73  
  presentation, 78  
  Web, 71, 78  
Timebox development, 18, 23, 41, 51  
TOAD, 306  
TopLink, 149  
Training  
  learn-as-you-go approach, 7  
  learning mediums, 9  
  software development requirements, 9  
  study groups, 10  
Transfer Object pattern, 77  

Tcl/Tk, 19, 24, 196  
Test Driven Development by Example (Beck), 339  
Test stubs, 67, 332
transient keyword, 76
TwoPi layout engine, 274
Two-tier architectures, 72
Type safety, 196

U
UML diagram
  use-case models, 40
UML diagrams
  activity, 92
  actors, 91
  architecture, views of, 88, 89
  class, 93
  deployment and component, 98
  interaction, 87, 94, 102
  listed, 89
  models, construction of low-cost, 86
  sequence and collaboration, 95
  statechart, 97
  use-case, 90
UML Distilled (Fowler), 111
UML2EJB, 178
Unified Modeling Language (UML)
  common vocabulary, benefits, 87
  executable UML, 108
  modeling tools, 102
  purpose of, 88, 89
  RUP, relationship to, 37, 40, 109
  standard, for modeling, 86
  stereotypes, 182
Unified Modeling Language User Guide, The (Booch), 111
Unit test bloat, 319
Unit tests, 341
Use case diagrams, 90
Use cases
  actors, 38
  architectural significance, 91
  defined, 38
  example, 39
flow of events section, 39, 93
primary, 91, 96
scenarios, 39
structure, 38
templates, 39, 58
UML diagram, 40, 91
validation, 87
Use-case model, 40
Use-case view, 89, 91, 96
User stories, 51

V
Validation, of design, 87, 96
Values Object pattern, 174
Vasseur, Alexandre, 246
Velocity Template Language (VTL), 116
  VTL references, 116
Veloedit, 302
Virtual teams, 87
Virus checkers, 269
Visitor pattern, 234
Visual development tools, 21
Vitruvius, 62
VTL directives, 116
VTL. See Velocity Template Language (VTL)

W
Waterfall lifecycle model
  case study, 31
  phases, 29, 41
  strengths and weaknesses, 30
Weaving, 236
Weaving methods, 237
Web applications. See Enterprise Web application
Web container, 78
Web Services Definition Language (WSDL), 124
Web tier, 71, 78
Web-centric architecture, 77
WebLogic Server, 264, 305, 351
WebLogic Workshop, 21
WebSphere, 251
White-box testing, 332, 342
Wiki, 13
Workflow detail, 45

X
XDoclet
AndroMDA, use with, 179
Ant build file, setting up, 125
Apache Ant, downloading, 125
boilerplate code generation, 125
class-level tags, 131
clean operations, implementing, 128
emergence of, 124
enterprise bean development, 125
file generation and output, 131
Hibernate plug-in, 163
installing, 125
Javadoc output, 125
Javadoc tags, 124
method-level tags, 131
output, separating source and generated, 128
reverse engineering, 134
servers supported, 129
session bean, creating, 129

Xerox Palo Alto Research Center (PARC), 231, 238, 244
XMI (XML Metadata Interchange), 180
XML, 68
XP (Extreme Programming)
code reviews, 53
contract constraints, 57
CRC (Class, Responsibilities, and Collaboration) card, 52
creation of, 48
design practices, 52
iteration plan, 50
J2EE Enterprise projects, using for, 56
key points, listed, 57
minimalist approach, 48
pair programming, 52
practices, twelve basic, 48
release plan, 50
roles, 54
scalability, 56
test-first development, 54, 318
timeboxed iterations, 51
user stories, 51
values, 49

Y
YAGNI, 69