# Index

abstract
class, 241
data type, 46
operation, 241
acceptance testing, 43
ACID transaction properties, 286
action, 20, 158
entry, 160
exit, 161
transition, 158
active object, 21, 53
activity diagram, 89
actor, 76
external system, 77
human, 77
input device, 77
primary, 76
secondary, 76
timer, 77
aggregate subsystem, 217
aggregation
hierarchy, 101
relationship, 17, 101
algorithm
class, 232
object, 128
analysis modeling, 63, 349
application
deployment, 314
logic object, 118, 127
architectural
communication patterns, 258
design, 41
association, 16, 95
class, 100, 271
many-to-many, 97
multiplicity of, 96
numerically specified, 96
one-to-many, 96
one-to-one, 96
optional, 96
ternary, 99
unary, 99
asynchronous message communication, 56, 227, 334
Pattern, 201
with Callback pattern, 259
atomic, 285
attribute, 47, 113
autonomy, 308
availability, 366
base use case, 82
black box testing, 64
boundary
class, 118, 119
object, 119
Broadcast Message Communication Pattern, 310
Broker
Forwarding Pattern, 280
Handle Pattern, 282
pattern, 280
business logic
class, 232, 239
object, 127
Call/Return pattern, 201
callback, 266
categorization of classes, 107, 116
Centralized Control Architectural Pattern, 320
choreography, 294
class, 46
design, 231–239
diagram, 16, 95
hierarchies, 52, 239
interface specification, 245
operations, 232–234
structuring, 117
structuring criteria, 116
client subsystem, 221
callout/server
architecture, 254
configuration, 260
systems, 253

© in this web service Cambridge University Press

www.cambridge.org
Client/Service architectural pattern, 254
CODARTS (Concurrent Design Approach for Real-Time Systems), 11
coding phase, 42
COMET (Collaborative Object Modeling and Architectural Design Method), 6, 61–64
communication diagram, 18, 133
component, 58, 194, 300
interface inheritance, 354
plug-compatible, 353
structuring criteria, 307
component-based software architecture, 194, 300, 353
component-based system, 58, 303
composite
component, 301, 306
object, 216
state, 163
subsystem, 217, 302
composition relationship, 17, 101
compound transaction, 288
pattern, 288
concurrent
application, 53
communication diagram, 21, 332, 338
object, 53
service design, 263
software architecture, 319
task, 322
condition, 19, 157
connector, 58, 305
constraint, 25, 103
control
object, 117, 124
subsystem, 224
task, 329
coordinator
object, 125
subsystem, 225
CORBA (Common Object Request Broker Architecture), 260, 286
data
abstraction, 48
abstraction class, 231, 234
flow oriented design, 8
structured design, 8
database wrapper class, 231, 266
demand driven
I/O task, 325
task, 328
deployment diagram, 23, 197
design
modeling, 63, 66, 353
pattern, 57
detailed design, 41, 246
device I/O boundary object, 120
discriminator, 102, 274
distributed application, 301
Distributed Control Architectural Pattern, 320
domain engineering, 345
dynamic
binding, 248
interaction modeling, 66, 132, 139, 349
modeling, 132
state machine modeling, 66, 151, 352
encapsulation, 48
entity
class, 104, 111, 231
object, 117, 123
entity-relationship (E-R) modeling, 10
task action, 20, 160
event, 19, 152
driven I/O task, 323
external, 336
internal, 336
sequencing logic, 338
synchronization, 336
timer, 336
evolutionary
dynamic analysis, 349
prototyping, 34
exit action, 20, 161
extend relationship, 85
extension
dss point, 86
use case, 85
external
class, 107
I/O device, 108
system, 108
timer, 108
user, 108
façade pattern, 263
feature, 528
alternative, 347
common, 346
optional, 347
feature modeling, 346
finite state machine, 151
foreign key, 269
functional requirement, 72
functional testing, 42, 64
generalization/specialization hierarchy, 17, 102,
239, 273
graphical user interaction (GUI) class, 231, 237
group message communication, 310
guard condition, 157
Hierarchical Control Architectural Pattern, 321
hierarchical statechart, 163
human actor, 77, 111
include relationship, 82
inclusion use case, 82
incremental
development, 34
software construction, 63
software integration, 64
information hiding, 9, 48, 230
class, 231
object, 48, 337
inheritance, 17, 52, 230, 239
input object, 121
input/output subsystem, 225
input task, 323–326
integrated communication diagram, 214
integration testing, 42
interaction diagram, 18, 132, 136
interface, 46, 206, 303
design, 206
provided, 304
required, 304
I/O task, 323–326
I/O subsystem, 225, 319
Jackson System Development, 9
Java, 249, 342
remote method invocation (RMI), 261
kernel system, 349
Layers of Abstraction pattern, 198
link, 95
location transparency, 280
long-living transaction, 288
loosely coupled message communication, 56, 334
maintainability, 357
message
communication, 226
dictionary, 216
sequence description, 138
sequence numbering, 136
middleware, 260
Model-Driven Architecture, 7
modifiability, 358
multicast communication, 311
Multiple Client/Multiple Service architectural pattern, 256
Multiple Client/Single Service architectural pattern, 254
Multi-tier Client/Service architectural pattern, 257
mutual exclusion, 54
Naval Research Lab Software Cost Reduction Method, 9
Negotiation pattern, 289–292
nodes, 23, 197, 201, 260, 300–302, 314–316
nonfunctional requirement(s), 73, 89, 357
object, 46
broker, 285
interaction modeling, 133–135
Modeling Technique, 10
structuring, 66, 117
Object Constraint Language (OCL), 103
Object Modeling Group (OMG), 3
object-based design, 10
object-oriented
analysis, 10
design, 10
OMG, 3, 6, 7
one-to-many association, 96, 271
one-to-one association, 96, 270
operation, 46, 47, 232
orchestration, 294
orthogonal statechart, 165
output object, 121
package, 21, 89
parent class, 52, 102, 239–241
passive object, 21, 53
performance, 309, 364
periodic
I/O task, 324
task, 327
physical class, 104
platform transparency, 280
polymorphism, 248
port, 304
postcondition in use case, 80
precondition in use case, 80
primary key, 269
producer/consumer, 54, 55
prototyping, 30, 34, 38, 40, 74
provided interface, 207, 297, 303–306
proxy object, 120
pseudocode, 246–249, 421–423
quality assurance, 40
real-time
software architecture, 318
system, 319
registration service, 284
relational
database, 268
table, 268
relationships, 95–102, 111–113, 123, 216, 232, 268
remote method invocation (RMI), 260–262
remote procedure call (RPC), 260
required interface, 207, 297, 303–306
requirements
analysis, 72
analysis & specification, 41
modeling, 63, 71, 72, 345
specification, 72
reusability, 363
risk analysis, 38, 65
ROOM, 11
scalability, 361
scenario, 135
scope of control, 219
security, 89, 365
separation of concerns, 216
sequence diagram, 19, 134
sequential
application, 53
service, 262
service design, 262
server, 253
service, 253, 279
class, 232
contract, 279
coordination, 294
Discovery Pattern, 282
Index

service (cont.)
  interface, 292
  object, 129
  Registration Pattern, 280
  reuse, 297
  subsystem, 223
service-oriented architecture (SOA), 278
  signature, 46
  simple component, 301, 306
  SOA (service-oriented architecture), 279–283,
    285–297
software,
  application engineering, 345
  architectural patterns, 198
  architecture, 5, 58, 193
  component, 300–310, 363
  deployment view, 197
  design concept, 5
  design method, 5
  design modeling, 67, 207–208
  design notation, 5
  dynamic view, 196
  life cycle, 29
  modeling, 3
  multiple views, 7, 194
  product line, 344
  product line engineering, 345
  product line evolution approach, 350
  quality assurance, 40
  quality attribute, 59, 357
  structural view, 195
  structuring criteria, 5
  system context class diagram, 123
  system context diagram, 105
  testing, 42
  validation, 40
  verification, 40
spiral model, 34, 65
  state, 19, 153
  transition, 19, 151
  statechart, 19, 152
  state-dependent
    control object, 125
    dynamic interaction modeling, 177
    stateless dynamic interaction modeling, 139
    state machine, 151–152, 177, 236–237
    state-machine class, 232, 226
  static, 94
  model, 94
  modeling, 66, 94, 349
  stereotype, 23, 107, 194
  subclass, 52, 102, 273
  Subscription/Notification Message
    Communication Pattern, 311
  substate, 163
  subsystem, 216
    communication diagram, 196, 216
    structuring criteria, 220
    superclass, 52, 102, 273
    synchronization, 54
    synchronous message communication, 227
    with reply, 56, 334
    with Reply pattern, 204, 258
    without reply, 57, 335
  system
    context diagram, 105
    deployment, 314
    testing, 42, 64
tagged value, 25
task, 21, 53, 208, 318–319
  behavior specification, 338
  communication, 332–336
  interface specification, 338
  structuring criteria, 323, 331
  testability, 360
  testing
    black box, 64
    integration, 42, 64
    functional, 42, 64
    system, 42, 64
    unit, 42, 63
    white box, 42, 64
  thread, 53, 342
  throwaway prototyping, 30
  tightly coupled message communication
    with reply, 36, 334
    without reply, 335
  timer event, 127, 156, 323–325, 327–328,
    336–337
  timer object, 126
  traceability, 360
  transaction, 285
  Two-Phase Commit Protocol, 286

UML (Unified Modeling Language), 6
  notation, 14
Unified Modeling Language (UML), 3–4, 14
Unified Software Development Process, 39, 64
  unit testing, 42
use case 74
  alternative sequence, 79
  diagram, 15
  main sequence, 79
  model, 74
  modeling, 65, 74, 345
  package, 89
  precondition, 80
  postcondition, 80
  user interaction
    object, 119
    subsystem, 221
    task, 330
  virtual interface, 48
  visibility, 17
  waterfall model, 29
  Web services, 283, 284
  white box testing, 42, 64
  white page brokering, 280
  whole/part relationship, 17, 100, 272
  World Wide Web, 200, 284, 424–425
  wrapper class, 266
yellow page brokering, 283
zero-or-one association, 96, 270