

Index

Note: Page numbers followed by *f* indicate figures and illustrations.

A

- abstract class
 - definition of, 10, 88n
 - generalization of, 88
 - source code for, 11–12
- Abstract Client*
 - in Broker Pattern, 398
 - in Data Bus Pattern, 380
 - in Observer Pattern, 372
 - in Proxy Pattern, 388–389
- Abstract Data* class, in Data Bus Pattern, 379–380
- abstract domains, 142–143, 143f, 144, 148
- Abstract Hardware Domain*, in Five-Layer Architecture Pattern, 150
- Abstract Instruction* class, in Virtual Machine Pattern, 178
- Abstract Interrupt Handler*, in Interrupt Pattern, 216
- abstract machines, in Virtual Machine Pattern, 176–177
- Abstract OS Domain*, in Five-Layer Architecture Pattern, 150
- Abstract Proxy*
 - in Broker Pattern, 398
 - in Proxy Pattern, 389–390
- Abstract Server*
 - in Broker Pattern, 398
 - in Proxy Pattern, 390
- Abstract Subject*
 - in Data Bus Pattern, 380
 - in Observer Pattern, 372–373
- Abstract Thread* class
 - in Critical Section Pattern, 310
 - in Cyclic Executive Pattern, 234
 - in Dynamic Priority Pattern, 253
 - in Highest Locker Pattern, 324–325
 - in Priority Ceiling Pattern, 331–333
 - in Priority Inheritance Pattern, 314–315
 - in Round Robin Pattern, 239
 - in Static Priority Pattern, 245
- Abstract Transformation* class, in Channel Architecture Pattern, 159
- abstraction, 21
 - for complex systems, 164, 173
 - in distribution architecture, 354
 - in physical architecture, 35f, 64–67
- «access» stereotype, 24
- action(s)
 - definition of, 8, 32
 - execution order of, 34
 - semantics of, 33
 - types of, 32
- action language, specifying, 33
- ActionSequence, 32
- «active» class
 - in components, 139
 - OS and, 92
- «active» objects
 - in architectural design, 123
 - in concurrency implementation, 92
 - rendezvous and, 92
 - semaphores for, 92
 - sizing, 31, 31f
 - strategies for reification of, 72–74

- «active» objects (*cont.*)
 - threads managed by, 139, 204–206, 205*f*
- activities
 - definition of, 33
 - in processes, 98, 99*f*
 - semantics of, 33
 - types of, 36
- activity diagrams
 - for algorithmic decomposition, 124
 - for architectural views, 67
 - for behavior, 121, 122, 124
 - function of, 6
 - notation for, 42*f*, 472
 - for requirements detailing, 120
 - for specification, 119
 - uses of, 41
- actors, 30–31, 30*f*, 49
- actual parameter, 22
- Actuation Channel(s)*
 - in Heterogeneous Redundancy Pattern, 429
 - in live-lock, 448
 - in Monitor-Actuator Pattern, 433
 - monitoring, 432, 438–439
 - in Safety Executive Pattern, 452
 - in Sanity Check Pattern, 439–440
 - in Triple Modular Redundancy Pattern, 423
 - in Watchdog Pattern, 445
- Actuation Data Source*
 - in Monitor-Actuator Pattern, 433
 - in Sanity Check Pattern, 440
 - in Watchdog Pattern, 446
- Actuation Validation* component
 - in Heterogeneous Redundancy Pattern, 427–428
 - in Homogenous Redundancy Pattern, 416
- Actuator* actor
 - in Heterogeneous Redundancy Pattern, 428–429
 - in Monitor-Actuator Pattern, 433
 - in Protected Single Channel Pattern, 410–411
 - in Safety Executive Pattern, 452
 - in Sanity Check Pattern, 440
 - in Triple Modular Redundancy Pattern, 423
 - in Watchdog Pattern, 446
- Actuator Monitor Sensor*
 - in Monitor-Actuator Pattern, 434
 - in Sanity Check Pattern, 440
- adapter pattern, 88
- advanced association, notation for, 460
- advanced sequence diagram, notation for, 464
- aggregation, 15–16
 - implementation of, 89
 - notation for, 15–16, 460
- algorithmic analysis, in System Engineering phase, 120
- algorithmic decomposition, activity diagrams for, 124
- allocation plan, in Static Allocation Pattern, 263
- analysis
 - definition of, 56
 - vs.* design, 118–119, 127
 - and functional requirements, 56
 - purpose of, 118
- analysis models
 - vs.* design models, 122
 - optimization of (*See* design patterns)
 - purpose of, 122
- analysis pattern, 127
- Analysis phase, 58–59, 112, 112*f*, 118–122
- and-states
 - concurrency with, 37
 - notation for, 470
 - vs.* or-states, 36, 37, 37*f*
 - synchronization of, 37–38
- API (application program interface), in Microkernel Architecture Pattern, 153
- application
 - creation of, with MDA, 85
 - portability of, 176–177
 - precompiling of, 182
- Application*, in Virtual Machine Pattern, 178
- Application Control Block*, in Virtual Machine Pattern, 178
- Application Domain*, in Five-Layer Architecture Pattern, 148
- application model, UML for, 6

- application program interface (API), in
Microkernel Architecture Pattern, 153
 - application protocol, 354
 - architectural design
 - architectural views in, 137–140
 - in design phase, 59
 - in Systems Engineering phase, 120
 - architectural design patterns
 - for architectural views elaboration, 123
 - scope of, 51–52, 128
 - Architectural Design phase
 - architectural views in, 123
 - integration test plan in, 125
 - architectural views
 - in architectural design, 123, 137–140
 - definition of, 52
 - elaboration of, 123
 - patterns and, 60
 - in ROPES process, 67, 68*f*
 - architecture
 - aspects of, 52
 - components as elements of, 70
 - definition of, 56
 - design and, 56
 - evaluation of, 118
 - extension of, for virtual machine,
181–182
 - integration and test of, 125
 - prototype focus on, 111
 - artifacts, in processes, 98–99, 99*f*
 - association(s), 13–15
 - advanced, notation for, 459
 - definition of, 13
 - in domain hierarchies, 62–63
 - implementation of, 17, 89–91, 90*f*
 - in layered architecture, 146
 - notation for, 13, 14*f*, 460
 - pointers for, 279
 - in ROOM Pattern, 197
 - types of, 13
 - association label, 13, 14*f*
 - asymmetric architecture
 - deployment diagrams for, 81
 - distribution implementation for, 92–93
 - objects in, 75
 - asymmetric distribution architecture,
354
 - asynchronous communication, in Message
Queuing Pattern, 208–209
 - asynchronous event transfer, implementa-
tion of, 36
 - asynchronous rendezvous, 92, 208–209
 - attributes
 - definition of, 7
 - in detailed design, 124
 - function of, 12
 - inheritance of, 20
 - in model execution, 105
- ## B
- «becomes» stereotype, 82
 - behavior
 - activity diagrams for, 121, 122
 - control law diagrams for, 121
 - sequence diagrams for, 122
 - statecharts for, 122
 - in UML, 84
 - behavioral aspect, of models, 6
 - behavioral design patterns, use of, 128
 - BehavioralFeatures, in UML metamodel,
32
 - «bind» stereotype, 21, 24
 - Binding dependency, 23. *See also* «bind»
stereotype
 - bit-dominance protocols, throughput of, 77
 - Blocked Queue*
 - in Dynamic Priority Pattern, 253–254
 - in Static Priority Pattern, 245
 - blocking, of tasks, 303–305, 304*f*. *See also*
chain blocking; priority inversion
 - Boundary Object*, in Guarded Call Pattern,
223
 - bounded priority inversion, 314, 323, 330
 - branch pseudostate, 38, 39*f*
 - Broker*, in Broker Pattern, 398
 - broker architecture, distributed objects in,
75
 - Broker Pattern, 395–402
 - abstract of, 395–396
 - application of, 395
 - collaboration roles in, 398–400
 - consequences in, 400, 401*f*
 - CORBA in, 83, 400–402, 401*f*

Broker Pattern (*cont.*)
 implementation of, 400–402
 patterns related to, 402
 problem addressed by, 396
 sample model of, 402, 402*f*–403*f*
 for “soft” real-time systems, 76–77
 structure of, 396, 397*f*
Buffered Ptr, in Garbage Compactor
 Pattern, 294–295

C

C programming language
 associations in, 89–91
 classes in, 86–87
 interfaces in, 87–88
 logical models in, 91
 member functions in, 86–87
 C++ programming language
 associations in, 89–91
 classes in, 86–87
 interfaces in, 87–88
 logical models in, 91
 name mangling in, 88–89
 polymorphism in, 88
 Pool Allocation Pattern in, 269–270
 «call» stereotype, 24
 CallAction, 32
 Callback object, in Rendezvous Pattern,
 229
 CallEvent, 34
 CAN bus protocol, 77, 355
Capsule, in ROOM Pattern, 194–195, 196
 CBD (Component-Based Development),
 28–29
 in component-based architecture,
 189–190
 CDMA (Collision Detect Multiple Access),
 77, 355
 chain blocking, 317, 319*f*, 323
 ChangeEvent, 34
 channel(s)
 in Channel Architecture Pattern, 159
 definition of, 157
 multiple (*See* multiple channels)
 parallel, 411, 421–422
 in Protected Single Channel Pattern, 411

 serial, 411
 single, for safety and reliability, 410
 Channel Architecture Pattern, 157–163
 abstract of, 157–158
 application of, 157
 collaboration roles in, 159–160
 consequences in, 160
 implementation of, 160–161
 patterns related to, 161
 problem addressed by, 158
 sample model of, 161–163, 162*f*
 structure of, 158–159, 159*f*
 checksum, for message integrity, 78
 choice point pseudostate, 39*f*, 40
 class(es). *See also* struct(s); subclasses;
 superclasses
 abstract (*See* abstract class)
 in C-based languages, 86–87
 vs. components, 28
 definition of, 8
 vs. interface, 12
 in layered architecture, 145
 in logical architecture, 138
 notation for, 458
 vs. objects, 6
 parameterized, 22, 458
 semantics of, 25
 in subsystems, 139
 in Subsystems and Components View,
 91
 class diagrams
 for architectural views, 67
 basic, 8, 9*f*
 collaborations in, 122
 for interfaces, 121
 notation for, 458–462
 purpose of, 24–25
 source code for, 10–12
 for structure, 123
 for subsystems (*See* subsystem diagram)
 classifiers
 notation for, 462
 in UML metamodel, 32
 client(s)
 data shared between, 377
 notification of, 370–371
 with unknown properties, 395–396

- Client actor*
 - in Fixed Size Buffer Plan, 276
 - in Garbage Collection Pattern, 288
 - in Garbage Compactor Pattern, 295
 - in Microkernel Architecture Pattern, 153
 - in Pool Allocation Pattern, 267
 - in Remote Method Call Pattern, 364–365
 - in Smart Pointer Pattern, 281
- Client Interface*, in Component-Based Architecture Pattern, 187
- Client Stub*, in Remote Method Call Pattern, 365
- Client Thread*
 - in Guarded Call Pattern, 223
 - in Rendezvous Pattern, 229
- Client-side Proxy*
 - in Broker Pattern, 398–399
 - in Proxy Pattern, 390
- closed layered architecture, classes in, 145
- collaboration(s). *See also* design patterns
 - in activity diagrams, 124
 - in analysis phase, limiting, 121–122
 - in class diagrams, 122, 124
 - definition of, 6, 51, 74, 135
 - in layered architecture, 146, 147f
 - in mechanistic design, 123
 - notation for, 131, 463
 - parameterized, 51, 131
 - in sequence diagrams, 122, 123, 124
 - in statecharts, 124
 - in System Engineering phase, 120
 - in thread reification, 74
 - for use cases, 121
- Collaboration*, in Component-Based Architecture Pattern, 187
- collaboration architecture, 354
- collaboration diagram
 - definition of, 43, 43f
 - function of, 6
 - notation for, 463
- Collectable* class, in Garbage Collection Pattern, 288
- Collision Detect Multiple Access (CDMA), 77
- COM+, 91–92, 189
- COM (Component Object Model), 189
- commercial component framework, 91–92
- Commercial Off The Shelf (COTS)
 - components, 189
- Commercial Off The Shelf (COTS) hardware, 66
- common mode fault, 408
- Common Warehouse Model (CWM), 85
- communication component, for
 - distribution implementation, 92–93
- Communication Domain*, in Five-Layer Architecture Pattern, 149–150
- communication infrastructure, purpose of, 356
- communication protocols
 - binding, in CORBA, 83–84
 - in distribution architecture, 354
 - throughput and, 77–78
- Comparator*, in Triple Modular Redundancy Pattern, 423
- “completion” event, 36
- component(s)
 - «active» classes in, 139
 - as architectural elements, 70
 - definition of, 6, 28, 184
 - in deployment models, 30
 - implementation of, 91–92
 - sizing, 31, 31f
 - in subsystem diagrams, 123
 - and subsystems, 28, 65f, 70–71, 139, 185
 - use of, 28–29
- Component*, in Component-Based Architecture Pattern, 187
- component architecture. *See* Subsystem and Component View
- component diagram
 - in architectural design, 123
 - notation for, 466
 - for Subsystem and Component View, 71, 71f
- Component Framework*, in Component-Based Architecture Pattern, 187
- Component Loader*, in Component-Based Architecture Pattern, 187
- Component Manager*, in Component-Based Architecture Pattern, 187
- Component Object Model (COM), 189
- Component Repository*, in Component-Based Architecture Pattern, 188

- Component View. *See* Subsystem and Component View
- Component-Based Architecture Pattern, 184–192
 - abstract of, 185–186
 - application of, 184–185
 - benefits of, 185
 - collaboration roles in, 187–188
 - consequences in, 188–189
 - implementation of, 189–190
 - patterns related to, 190–191
 - problem addressed by, 186
 - sample model of, 191–192, 191*f*, 193*f*
 - structure of, 186*f*
- Component-Based Development (CBD), 28–29
 - in component-based architecture, 189–190
- composite objects, in Static Allocation Pattern, 263
- composition, 16–17
 - in Hierarchical Control Pattern, 174
 - implementation of, 89
 - notation for, 14*f*, 16–17, 460
 - of passive objects, 139, 204, 205*f*
 - in Subsystems and Components View, 91
- concept reuse, with patterns, 126
- Concrete Client*
 - in Broker Pattern, 399
 - in Data Bus Pattern, 381
 - in Observer Pattern, 373
 - in Proxy Pattern, 390
- Concrete Data*, in Data Bus Pattern, 380
- Concrete Instruction* class, in Virtual Machine Pattern, 178
- Concrete Interrupt Handler*, in Interrupt Pattern, 216
- Concrete Server*
 - in Broker Pattern, 399
 - in Proxy Pattern, 390
- Concrete Subject*
 - in Data Bus Pattern, 381
 - in Observer Pattern, 373
- Concrete Thread*
 - in Critical Section Pattern, 310
 - in Cyclic Executive Pattern, 234
 - in Dynamic Priority Pattern, 254
 - in Highest Locker Pattern, 325
 - in Priority Ceiling Pattern, 333
 - in Priority Inheritance Pattern, 315
 - in Round Robin Pattern, 239
 - in Static Priority Pattern, 245
- Concrete Transformation* class, in Channel Architecture Pattern, 160
- concurrency
 - with and-states, 37
 - definition of, 72
 - implementation of, 92
 - management of, 204–206
 - modeling, 72
 - in physical architecture abstraction, 66
 - in sequence diagrams, 44–47, 46*f*
 - threads in, 204–206, 205*f*
- Concurrency and Resource View
 - in architectural design, 123, 139
 - implementation of, 92
 - purpose of, 72, 139
 - task diagram for, 72, 73*f*
 - thread reification in, strategies for, 72–74
- concurrency architecture. *See* Concurrency and Resource View
- concurrency patterns, 204–206
- conditional pseudostate, 38, 39*f*
- Configuration Items, packages as, 26
- configuration management, defining, 117
- Connector*, in ROOM Pattern, 196
- Consequences, in design patterns, 51, 59, 130
- constraints
 - notation for, 462
 - for requirements detailing, 120
 - in System Engineering phase, 121
- container class, for association implementation, 89–91, 90*f*
- Control Interface*, in Hierarchical Control Pattern, 170–171, 172
- control law diagrams
 - for behavior, 121
 - for requirements detailing, 120
- control law specification, in System Engineering phase, 120
- Controller* role, in Hierarchical Control Pattern, 172

- CORBA (Common Object Request Broker Architecture)
 - binding with, 83
 - in Broker Pattern, 83, 400–402, 401*f*
 - for component implementation, 189
 - definition of, 83
 - core services, 151–152, 155
 - COTS (Commercial Off The Shelf)
 - components, 189
 - COTS hardware, 66
 - «create» stereotype, 24
 - CreateAction, 32
 - Critical Section Pattern, 308–313
 - abstract of, 308–309
 - application of, 308
 - collaboration roles in, 310–311
 - consequences in, 311
 - implementation of, 311–312
 - patterns related to, 312
 - problem addressed by, 309
 - sample model of, 312–313, 312*f*–313*f*
 - structure of, 309*f*
 - criticality, 242
 - CWM (Common Warehouse Model), 85
 - Cyclic Executive Pattern, 232–237
 - abstract of, 232–233
 - application of, 232
 - collaboration roles in, 234
 - consequences in, 234–235
 - implementation of, 235–236
 - patterns related to, 236–237, 236*f*
 - problem addressed by, 233
 - structure of, 233, 233*f*
 - cyclic redundancy check (CRC), for
 - message integrity, 78, 355
- D**
- dangling pointers, 280
 - Data Bus*, in Data Bus Pattern, 381–382
 - Data Bus Pattern, 377–387
 - abstract of, 377
 - application of, 377
 - collaboration roles in, 379–382
 - consequences in, 383
 - implementation of, 383
 - patterns related to, 383–384
 - problem addressed by, 377
 - sample model of, 384–387, 385*f*, 386*f*
 - structure of, 377–378, 378*f*, 379*f*
 - Data class*
 - in Broker Pattern, 399
 - in Observer Pattern, 373
 - in Proxy Pattern, 390
 - Data Client* instance, in Shared Memory Pattern, 357
 - Data ID: ID Type* attribute, in Data Bus Pattern, 380
 - Data Integrity Checks*
 - in Monitor-Actuator Pattern, 434
 - in Sanity Check Pattern, 440
 - Data Name: String* attribute, in Data Bus Pattern, 380
 - data sharing, between processors, 356–357
 - Data Source* instance, in Shared Memory Pattern, 358
 - data stream. *See* channel(s)
 - Data Transformation* component
 - in Heterogeneous Redundancy Pattern, 429
 - in Homogenous Redundancy Pattern, 417
 - in Monitor-Actuator Pattern, 434
 - in Protected Single Channel Pattern, 411–412
 - in Safety Executive Pattern, 452
 - in Sanity Check Pattern, 440
 - in Triple Modular Redundancy Pattern, 423
 - in Watchdog Pattern, 447
 - Data Transport Domain*, in Five-Layer Architecture Pattern, 149–150
 - Data Type*, in Data Bus Pattern, 382
 - Data Units* attribute, in Data Bus Pattern, 380
 - Data Validation* component
 - in Heterogeneous Redundancy Pattern, 429
 - in Homogenous Redundancy Pattern, 417
 - in Protected Single Channel Pattern, 412
 - datum
 - in Channel Architecture Pattern, 160
 - in Triple Modular Redundancy Pattern, 424

- DCOM, 189
- deadlock
 - avoiding, 330, 338, 345
 - definition of, 305
 - detecting, 447–448
 - in resource management, 305–308, 307*f*
- debugging
 - in model execution, 103–104, 104*f*
 - in virtual machines, 183
- decomposition
 - algorithmic, activity diagrams for, 124
 - hardware-software, deployment diagram for, 121
 - Recursive Containment Pattern for, 164
- deep history pseudostate, 39*f*, 40
- default pseudostate, 39*f*, 40
- defects
 - strategic, expense of, 106
 - in waterfall lifecycle, 110
- dependency, 23*f*
 - in domain hierarchies, 62–63
 - types of, 21
- deployment, prototype focus on, 111
- deployment diagram
 - in architectural design, 123
 - for asymmetric architecture, 81
 - for Deployment View, 81, 82*f*
 - for hardware-software decomposition, 121
 - notation for, 466
 - for symmetric architecture, 81
- deployment model, 30
- Deployment View
 - in architectural design, 123, 139
 - asymmetric architecture in, 81
 - deployment diagram for, 81, 82*f*
 - implementation of, 93
 - purpose of, 81, 139
- design
 - vs.* analysis, 118–119, 127
 - in analysis phase, 121–122
 - and architecture, 56
 - prototype focus on, 111
 - qualities of service and, 56
- design automation tools, *vs.* drawing tools, 7
- design faults. *See* systematic faults
- design model, *vs.* analysis model, 122
- design patterns. *See also* frameworks; pattern *entries*
 - vs.* analysis model, 50
 - architectural (*See* architectural design patterns)
 - aspects of, 51, 129–130
 - concept of, 126
 - definition of, 50, 59, 127, xix
 - evaluation of, 133
 - index of, 473–474
 - instantiation of, 133, 134–135, 135*f*
 - mechanistic, 51–52, 123
 - mixing, 60
 - as parameterized collaborations, 51
 - qualities of service in, 50–51, 59
 - reading, 130
 - structure of, 129–130
 - testing, 133
- Design phase, 59, 112–113, 112*f*
- DestroyAction, 32
- Detailed Design phase, 124
- «device» node stereotype, 30
- diagrams, mission of, 25
- directionality, notation for, 14*f*, 15
- distribution architecture. *See* Distribution View
- Distribution View, 76*f*
 - abstraction in, 354
 - in architectural design, 123, 139–140
 - asymmetric, 354
 - implementation of, 92–93
 - importance of, 354
 - performance in, 75–77
 - purpose of, 75, 139
 - reliability in, 78–79
 - selection of, 75
 - symmetric, 354
 - throughput in, 77–78
- diverse redundancy. *See* heterogeneous redundancy
- domain(s)
 - abstract, 142–143, 143*f*, 144, 148
 - construction of, in Recursive Containment Pattern, 166
 - definition of, 60, 138
 - generalization in, 63

- hierarchical, 62–63, 62*f*
 - in logical architecture, 60–63, 138
 - modeling, 138
 - in object analysis, 121
 - vs.* packages, 60
 - in physical architecture, 63, 64*f*
 - purpose of, 138
 - in subsystems, 138
 - domain diagrams, for logical architecture, 60–62, 61*f*
 - domain model, of layered architecture, 146, 147*f*
 - domain* role
 - in 5-Layered Architecture Pattern, 149–150
 - in Layered Pattern, 145
 - «domain» stereotype
 - in logical architecture, 60, 138
 - in logical model implementation, 91
 - mission of, 60
 - in package use, 26
 - Domain View, in architectural design, 138
 - drawing tools, *vs.* design automation tools, 7
 - dynamic memory allocation
 - difficulties with, 260–262
 - fragmentation in, 261–262, 273–274
 - dynamic model-code associativity, in ROPES process, 105
 - Dynamic Priority Pattern, 251–257
 - abstract of, 251–252
 - application of, 251
 - collaboration roles in, 252–255
 - consequences in, 255–256
 - implementation of, 256
 - patterns related to, 256
 - problem addressed by, 252
 - sample model of, 256–257, 257*f*
 - structure of, 252, 253*f*
- E**
- EJB (Enterprise Java Beans), 189
 - Element*, in Recursive Containment Pattern, 164
 - End Port*, in ROOM Pattern, 196
 - engineering approach, selection of, 117
 - Enterprise Java Beans (EJB), 189
 - entry actions, guards and, 36
 - errors. *See* systematic faults
 - essential properties, 118
 - event(s)
 - interrupts for, 214
 - parameters of, 35–36
 - types of, 34
 - event loop, in Cyclic Event Pattern, 233
 - event source strategy, for thread reification, 73
 - event-name, 34
 - Example, in design patterns, 51
 - executing model. *See* model execution
 - exit actions, guards and, 36
 - extends, in use cases, 48
 - extension
 - of architecture, 181–182
 - definition of, 20–21
 - of frameworks, 129
 - External Services component, in Microkernel Architecture Pattern, 154
 - Extreme Programming (XP) approach, in nanocycle, 112
- F**
- Façade Pattern, 11
 - Fail-safe Processing Channel*, in Safety Executive Pattern, 452
 - fail-safe state, 406–409
 - failures. *See* random faults
 - faults
 - handling, 408–409
 - redundancy and, 408
 - transient, 413, 436
 - types of, 406
 - feedback fault correction, 408
 - feedforward fault correction, 408–409
 - final pseudostate, 38–39, 39*f*
 - finite state machine (FSM), 34
 - Five-Layer Architecture Pattern, 148–151
 - abstract of, 148
 - application of, 148
 - collaboration roles in, 148–150
 - consequences in, 150
 - problem addressed by, 148

- Five-Layer Architecture Pattern (*cont.*)
 - sample model of, 150–151, 151*f*
 - structure of, 148, 149*f*
 - Fixed Size Buffer Plan, 273–278
 - abstract for, 273
 - application of, 273
 - collaboration roles in, 276
 - consequences in, 277
 - implementation of, 277
 - patterns related to, 277
 - problem addressed by, 274
 - sample model of, 277, 278*f*
 - structure of, 274–275, 275*f*
 - fork pseudostate, 39, 39*f*
 - formal behavioral specification language, 49
 - formal parameter, 21–22
 - Formatter*, in Remote Method Call Pattern, 365
 - fragmentation. *See* memory fragmentation
 - frameworks
 - commercial, 109
 - definition of, 109, 128
 - disadvantages of, 129
 - purpose of, 109
 - in ROPES process, 109
 - use of, 128–129
 - Free Block List*
 - in Fixed Size Buffer Plan, 276
 - in Garbage Collection Pattern, 288
 - in Garbage Compactor Pattern, 295
 - «friend» stereotype, 24
 - FSM (finite state machine), 34
 - Functional Interface*, in Hierarchical Control Pattern, 170–171, 172
 - functional requirements
 - analysis and, 56
 - definition of, 47
 - in UML, 84
- G**
- Garbage Collection Pattern, 286–292
 - abstract of, 286
 - application of, 286
 - collaboration roles in, 288
 - consequences in, 288–289
 - implementation of, 289–290
 - patterns related to, 290
 - problem addressed by, 286–287
 - sample model of, 290–292, 291*f*, 292*f*
 - structure of, 287, 287*f*
 - Garbage Collector*, in Garbage Collection Pattern, 288
 - Garbage Compactor*, in Garbage Compactor Pattern, 295
 - Garbage Compactor Pattern, 293–299
 - abstract of, 293
 - collaboration roles in, 294–296
 - consequences in, 296–297
 - implementation of, 297
 - patterns related to, 297
 - problem addressed by, 293
 - sample model of, 297–299, 298*f*, 299*f*
 - structure of, 293–294, 294*f*
 - generalization, 22*f*
 - of abstract classes, 88
 - definition of, 48
 - in domains, 63
 - of frameworks, 128–129
 - function of, 18
 - notation for, 461
 - virtual methods for, 89
 - Generic Pool Manager*, in Pool Allocation Pattern, 267–268
 - Global Data* object, in Shared Memory Pattern, 358
 - guard
 - for and-state synchronization, 37–38
 - definition of, 36
 - Guarded Call Pattern, 221–227
 - abstract of, 221–222
 - application of, 221
 - collaboration roles in, 223–224
 - consequences in, 224
 - implementation of, 224–225
 - patterns related to, 225
 - problem addressed by, 222
 - sample model for, 225–227, 226*f*
 - structure of, 222, 222*f*
- H**
- Hamming codes, for message integrity, 78–79, 78*f*

- hard real-time systems, performance in, 75–76
 - hardware
 - in Protected Single Channel Pattern, 410
 - in Shared Memory Pattern, 356, 359
 - Hardware Semaphore*, in Shared Memory Pattern, 358–359
 - Heap*
 - in Garbage Collection Pattern, 288
 - in Garbage Compactor Pattern, 295
 - Heap Manager*, in Fixed Size Buffer Plan, 276
 - heterogeneous redundancy, 79, 80f
 - Heterogeneous Redundancy Pattern, 426–431
 - abstract of, 426–427
 - application of, 426
 - collaboration roles in, 427–429
 - consequences in, 429–430
 - implementation of, 430
 - patterns related to, 430
 - problem addressed by, 427
 - sample model of, 430–431, 431f
 - structure of, 427, 428f
 - Hierarchical Control Pattern, 170–176
 - abstract of, 170–171
 - application of, 170
 - collaboration roles in, 172–173
 - consequences in, 173
 - implementation of, 173–174
 - patterns related to, 174
 - problem addressed by, 171
 - sample model of, 175–176, 175f
 - structure of, 171, 172f
 - Highest Locker Pattern, 323–328
 - abstract of, 323
 - application of, 323
 - collaboration roles in, 324–326
 - consequences in, 326–327
 - implementation of, 327
 - patterns related to, 328
 - problem addressed by, 323
 - sample model of, 328, 329f
 - structure of, 323–324, 324f
 - Homogenous Redundancy Pattern, 415–421
 - abstract of, 416
 - collaboration roles in, 416–418
 - consequences in, 418–419
 - faults in, 415, 418–419
 - implementation of, 419
 - patterns related to, 419
 - problem addressed by, 416
 - sample model of, 419–421, 420f
 - structure of, 416, 417f
- ## I
- ID Type*, in Data Bus Pattern, 382
 - IDL (Interface Description Language)
 - in Broker Pattern, 400–402, 401f
 - in component implementation, 190
 - use of, 84
 - implementation, prototype focus on, 111
 - implementation diagrams, notation for, 466
 - «import» stereotype, 24
 - includes, in use cases, 48
 - Info Type: Data Type* attribute, in Data Bus Pattern, 380
 - infrastructure standards, 84. *See also* CORBA
 - inheritance
 - of attributes, 20
 - in generalization, 18
 - initial pseudostate, 39f, 40
 - Input Filter* class, in Channel Architecture Pattern, 160
 - Input Processing* component
 - in Heterogeneous Redundancy Pattern, 429
 - in Homogenous Redundancy Pattern, 417
 - in Protected Single Channel Pattern, 412
 - in Safety Executive Pattern, 452
 - in Triple Modular Redundancy Pattern, 423
 - Input Sensor* actor
 - in Heterogeneous Redundancy Pattern, 429
 - in Homogenous Redundancy Pattern, 418
 - in Protected Single Channel Pattern, 412
 - in Safety Executive Pattern, 452
 - in Triple Modular Redundancy Pattern, 423

- instance, 8, 139
 - instance multiplicity, 17
 - «instantiate» stereotype, 24
 - instantiation
 - of frameworks, 128
 - of patterns, 133
 - Instruction Engine*, in Virtual Machine Pattern, 178
 - integration and test phase, 59. *See also* Test phase; Translation phase
 - integration plan, for architecture testing, 125
 - Integrity Checks* component, in Watchdog Pattern, 447
 - interaction(s)
 - definition of, 41–43
 - function of, 6
 - ordering, 44–47, 46f
 - interaction diagrams, 6
 - interaction protocols, in System Engineering phase, 120
 - interface(s)
 - vs.* class, 12
 - class diagrams for, 121
 - in component-based architecture, 187, 188
 - definition of, 8
 - in Hierarchical Control Pattern, 170–171, 174
 - implementation of, 87–88
 - management of, in Component-Based Architecture Pattern, 188
 - in ROOM Pattern, 194–195, 196–197
 - in Subsystems and Components View, 91
 - interface class, as interface substitute, 88
 - interface compliance
 - generalization for, 18
 - interfaces for, 10
 - Interface Description Language. *See* IDL
 - interface device strategy, for thread reification, 74
 - Interface Pattern, 11
 - Internal Services* component
 - implementation of, 155
 - in Microkernel Architecture Pattern, 154
 - interrupt handlers. *See* Abstract Interrupt Handler; Concrete Interrupt Handler
 - Interrupt Pattern, 214–220
 - abstract of, 214
 - application of, 214
 - collaboration roles in, 216–217
 - consequences in, 217–218
 - implementation of, 218–219
 - patterns related to, 219–220
 - problem addressed by, 214
 - sample model of, 220, 220f–221f
 - structure of, 214–215, 215f, 216f
 - Interrupt Vector*, in Interrupt Pattern, 217
 - Interrupt Vector Table*, in Interrupt Pattern, 217
 - ISO communications model, protocol stack in, 354
 - iterative development, 107–110, 108f. *See also* ROPES; spiral development lifecycle
 - iterative prototypes
 - focus of, 111
 - in microcycle, 111
 - in spiral lifecycle, 110–111
- J**
- Java programming language
 - associations in, 89–91
 - classes in, 86
 - interfaces in, 87
 - logical models in, 91
 - Pool Allocation Pattern in, 270–271
 - JINI, 189
 - join pseudostate, 39f, 40
 - junction pseudostate, 39f, 40
 - Just-In-Time (JIT) compilation, 183
- K**
- Keyed Watchdog, 448
- L**
- latent fault, 436
 - layered architecture
 - associations in, 146
 - classes in, 145
 - collaborations in, 146, 147f
 - domain model of, 146, 147f
 - portability of, 144, 145, 148

- layered models, for application creation, 85
 - Layered Pattern, 142–146
 - abstract of, 142–144
 - collaboration roles in, 145
 - consequences in, 145
 - function of, 142
 - implementation of, 146
 - patterns related to, 146
 - problem addressed by, 144
 - sample model of, 146, 147*f*
 - structure of, 144, 144*f*
 - for virtual machine, 181
 - Leaf Element* class, in Hierarchical Control Pattern, 172–173
 - legacy systems, MDA compliance of, 86
 - lifelines. *See also* thread(s)
 - ordering interactions and, 46–47, 46*f*
 - in sequence diagrams, 44
 - Listener* object, in Data Bus Pattern, 382
 - live-lock, 448
 - local format, in communication
 - infrastructure, 356
 - Local Notification Handle* class
 - in Broker Pattern, 399
 - in Proxy Pattern, 391
 - logical architecture
 - definition of, 58
 - domains in, 60–63
 - vs.* physical architecture, 58, 58*f*
 - in ROPES process, 60
 - logical model
 - focus of, 138
 - implementation of, packages in, 91
 - and physical model, separation of, 139
 - lollipop notation, 9–10
- M**
- macrocycle, 110–111, 111*f*
 - macrophases, 110–111
 - Management Interface*, in Component-Based Architecture Pattern, 188
 - MDA (Model-Driven Architecture)
 - flexibility of, 86
 - purpose of, 82–83, 85
 - mean time between failure (MTBF), 79
 - mechanistic design patterns, 123
 - scope of, 51–52, 128
 - Mechanistic Design phase, 123–124
 - member functions, and structs, 86–87
 - memory allocation
 - dynamic (*See* dynamic memory allocation)
 - static (*See* Static Allocation Pattern)
 - Memory Block*
 - in Garbage Collection Pattern, 288
 - in Garbage Compactor Pattern, 295
 - memory fragmentation
 - in dynamic memory allocation, 261–262, 273–274
 - removal of, 293
 - memory leaks
 - elimination of, 286–287
 - and pointers, 280
 - Memory Segment*, in Fixed Size Buffer Plan, 276
 - merge junction pseudostate, 39*f*, 40
 - message integrity, 78–79, 355
 - Message* object, in Shared Memory Pattern, 359
 - Message Queue*, in Shared Memory Pattern, 359
 - Message Queuing Pattern, 207–213
 - abstract of, 208–209
 - application of, 207
 - collaboration roles in, 210–211
 - consequences in, 211
 - implementation of, 211
 - patterns related to, 212
 - problem addressed by, 209
 - sample model of, 212–213, 213*f*
 - structure of, 209–210, 210*f*
 - message syntax, notation for, 463
 - Metamodel Object Facility (MOF), 85
 - methods
 - definition of, 7, 33
 - function of, 12
 - vs.* operations, 33
 - in subclasses, 19
 - in UML metamodel, 32
 - MFC (Microsoft Foundation Class), 109
 - microcycle
 - microphases of, 112–113, 112*f*, 116, 117*f*
 - (*See also specific phases*)

- in ROPES process, 111
 - Microkernel Architecture Pattern, 151–157
 - abstract of, 152
 - application of, 151
 - collaboration roles in, 153–154
 - consequences in, 154
 - implementation of, 154
 - patterns related to, 155–156
 - problem addressed by, 152
 - sample model of, 156–157, 156f
 - structure of, 152, 153f
 - for virtual machine, 181
 - Microkernel component
 - implementation of, 155
 - in Microkernel Architecture Pattern, 154
 - microphases, 112–113, 112f, 116, 117f
 - Microsoft Foundation Class (MFC), 109
 - Middleware Domain, in Five-Layer Architecture Pattern, 149–150
 - middleware standards, 84. *See also* CORBA
 - model(s). *See also* visual modeling; *specific models*
 - definition of, 5–6
 - and diagrams, 24
 - of Operations, 33
 - semantics of, 25
 - and source code, association of, 105–106
 - model execution
 - debugging and testing in, 103–104, 104f
 - purpose of, 102–103
 - in ROPES process, 101–105, 102f
 - simplicity of, 105
 - vs.* traditional development, 103
 - visual debugger in, 103
 - model-code associativity, in ROPES process, 105–106
 - Model-Driven Architecture (MDA)
 - flexibility of, 86
 - purpose of, 82–83, 85
 - ModelElements, in UML metamodel, 32
 - modeling standard. *See* UML
 - MOF (Metamodel Object Facility), 85
 - Monitor
 - in Monitor-Actuator Pattern, 434
 - in Sanity Check Pattern, 440
 - Monitor-Actuator Pattern, 432–438
 - abstract of, 432
 - application of, 432
 - collaboration roles in, 433–435
 - consequences in, 435
 - implementation of, 435–436
 - patterns related to, 436
 - problem addressed by, 432
 - sample model of, 437–438, 437f
 - structure of, 433, 433f
 - Monitoring Channel, in Monitor-Actuator Pattern, 434
 - Monitoring Input Processing
 - in Monitor-Actuator Pattern, 434
 - in Sanity Check Pattern, 440
 - monitoring personnel, in fault handling, 408–409
 - MTBF (mean time between failure), 79
 - Multidisciplinary Level, in physical architecture abstraction, 65f, 66
 - multiple channels
 - coordinating, 450–451
 - for safety and reliability, 415–416
 - for systematic fault detection, 426–427
 - for throughput efficiency, 158
 - multiplicity, 14–15, 14f
 - MultiResource object, in Simultaneous Locking Pattern, 339–340
 - Mutex semaphore
 - in Dynamic Priority Pattern, 254
 - in Guarded Call Pattern, 222, 223
 - in Highest Locker Pattern, 325
 - in Message Queuing Pattern, 210–211
 - in Ordered Locking Pattern, 347
 - in Priority Ceiling Pattern, 333
 - in Priority Inheritance Pattern, 315–316
 - in Simultaneous Locking Pattern, 340
 - in Static Priority Pattern, 245–246
 - mutual exclusion problem, 207–208, 208f. *See also* Mutex semaphore; PartLock semaphore
- N**
- name, of design patterns, 129
 - name mangling, 88–89
 - name scoping operator, 63
 - namespace, 91
 - nanocycle

- activities of, 113, 113*f*
- goal of, 113
- object model evaluation in, 122
- purpose of, 111–112
- in ROPES process, 111–112
- scope of, 111
- nested states, notation for, 468
- network format, in communication
 - infrastructure, 356
- nodes, 30
 - definition of, 30
 - purpose of, 81
 - stereotypes of, 81
- notes, notation for, 462
- Notification Handle* class
 - in Broker Pattern, 399
 - in Data Bus Pattern, 382
 - in Observer Pattern, 373
 - in Proxy Pattern, 391
- Notification List*, in Data Bus Pattern, 382
- “null” event, 36

O

- object(s)
 - active (*See* «active» objects)
 - in asymmetric architecture, 75
 - attributes of (*See* attributes)
 - vs.* classes, 6
 - composite, in Static Allocation Pattern, 263
 - definition of, 7
 - notation for, 458
 - in symmetric architecture, 75
- object analysis, purpose of, 121
- Object Analysis phase, 121–122
- Object Broker, purpose of, 83, 395–396
- object diagrams, use of, 25
- Object Factory*, in Fixed Size Buffer Plan, 276
- Object Level, in physical architecture
 - abstraction, 66–67
- Object Management Group (OMG), 4
- object model, evaluation of, in nanocycle, 122
- object pointer, 17
- object reference, 17
- Object Request Brokers (OBJ), 400
- Object Windows Library (OWL), 109
- Observer Pattern, 370–375
 - abstract of, 370–371
 - application of, 370
 - collaboration roles in, 372–373
 - consequences in, 373–374
 - implementation of, 374–375
 - patterns related to, 375
 - problem addressed by, 371
 - sample model of, 375, 376*f*
 - structure of, 371, 372*f*
- OBJ (Object Request Brokers), 400
- OMG (Object Management Group), 4
- open layered architecture, classes in, 145
- operating system, and «active» classes, 92
- operations
 - definition of, 33
 - vs.* Methods, 33
 - modeling, 33
 - in UML metamodel, 32
- optimality, 242–243
- optimization
 - of analysis models (*See* design patterns)
 - and deoptimization, 126–127
- Ordered Locking Pattern, 345–351
 - abstract of, 345
 - application of, 345
 - collaboration roles in, 347–348
 - consequences in, 348
 - implementation of, 348–349
 - patterns related to, 349
 - problem addressed by, 345
 - sample model of, 349, 350*f*–351*f*
 - structure of, 345–347, 346*f*
- or-states, *vs.* and-states, 36, 37, 37*f*
- orthogonal substates. *See* and-states
- Output Filter* class, in Channel Architecture Pattern, 160
- Output Processing* component
 - in Heterogeneous Redundancy Pattern, 429
 - in Homogenous Redundancy Pattern, 418
 - in Monitor-Actuator Pattern, 434
 - in Protected Single Channel Pattern, 412
 - in Safety Executive Pattern, 453
 - in Sanity Check Pattern, 441

Output Processing component (*cont.*)
 in Triple Modular Redundancy Pattern, 423
 in Watchdog Pattern, 447
 OWL (Object Windows Library), 109

P

package(s)
 contents of, 26
 definition of, 26
 domain role as, 145
 vs. domains, 60
 in logical model implementation, 91
 notation for, 26, 27*f*
 semantics of, 26
 use of, 26
 package diagram, notation for, 467
 parallel channels, 411, 421–422
 parameter(s)
 for collaborations, 51
 of events, 35–36
Parameter class, in Virtual Machine Pattern, 178
 parameterized class
 definition of, 22
 notation for, 458
 parameterized collaborations
 design patterns as, 51
 notation for, 131
 parameterized frameworks, 129
 parity bit, for message integrity, 78
 part object(s), composite and, 16–17
Part Object, in Static Allocation Pattern, 263
PartLock semaphore, in Simultaneous Locking Pattern, 340
 Party phase, 112, 112*f*, 116–117
 passive objects, composition of, 139, 204, 205*f*
 pattern hatching, 126, 132–133, 132*f*
 pattern matching, 133
 pattern mining, 126, 134, 134*f*
 PDA (platform-dependent application), 85
 performance, in Distribution View, 75–77
 Permission dependency, 24

physical architecture
 abstraction in, 35*f*, 64–67
 domain structure in, 63, 64*f*
 elements in, 64
 focus of, 63
 vs. logical architecture, 58, 58*f*
 model organization for, 67
 in ROPES process, 67
 physical model
 implementation of, 91
 and logical model, separation of, 139
 PIM (platform independent model), 85
Platform
 in Microkernel Architecture Pattern, 154
 virtual machine and, 180–182
 platform independence, 85
 platform independent model (PIM), 85
 platform-dependent application (PDA), 85
 pointer(s)
 for member functions, 87
 problems with, 278–280
 pointer arithmetic defects, 280
 polymorphism
 in C++, 88
 definition of, 18
 notation for, 18–19, 19*f*
 Pool Allocation Pattern, 266–273
 abstract of, 266
 application of, 266
 collaboration roles in, 267–268
 consequences in, 268–269
 implementation of, 269–271
 patterns related to, 271
 problem addressed by, 266
 sample model of, 271–273, 272*f*
 structure of, 266–267, 267*f*
PooledClass, in Pool Allocation Pattern, 268
Port, in ROOM Pattern, 195, 196
Port Mapper, in Remote Method Call Pattern, 365
 portability
 of applications, 176–177
 in layered architecture, 144, 145, 148
 postconditional invariants, 9
 precondition, 228
 preconditional invariants, 8
Primary Actuation Channel, 418, 429, 448

- Primary Actuation Validation* component, 427–428
- Primary Actuator* actor, 417, 418, 428–429
- Primary Data Transformation* component, 429
- Primary Data Validation* component, 429
- Primary Input Processing* component, 429
- Primary Input Sensor* actor, 418, 429
- Primary Output Processing* component, 429
- Primitive Object*, 263
- Priority Ceiling Pattern, 330–338
 - abstract of, 330
 - application of, 330
 - collaboration roles in, 331–334
 - consequences in, 334–335
 - implementation of, 335
 - patterns related to, 335
 - problem addressed by, 330
 - sample model of, 335–338, 336f–337f
 - structure of, 330, 331f, 332f
- Priority Inheritance Pattern, 314–322
 - abstract of, 314
 - application of, 314
 - collaboration roles in, 314–317
 - consequences in, 317–318, 318f, 319f
 - implementation of, 320
 - patterns related to, 320
 - problem addressed by, 314
 - sample model of, 320–322, 321f
 - structure of, 314, 315f
- priority inversion. *See also* task(s), blocking
 - bounded, 314, 323, 330
 - definition of, 305
 - unbounded, 305, 306f, 342
- priority scheduling. *See also* Scheduler
 - dynamic, 251–252
 - static, 242–243, 303
 - for urgency and criticality management, 242, 251
- priority-based protocols, 77–78, 355
- Problem, in design patterns, 51, 59, 129–130
- process
 - benefits of, 97–98
 - definition of, 98
 - elements of, 98–99, 99f
 - purpose of, 97
- Process Assessment, in Party phase, 118
- processing, monitoring, 443–444
- processor(s), data shared between, 356–357, 377
- Processor*, in Shared Memory Pattern, 359
- «processor» node stereotype, 30
- profiles, 84–85
- propagation of events, for and-state synchronization, 38
- Protected Single Channel Pattern, 409–415
 - abstract of, 410
 - application of, 409–410
 - collaboration roles in, 410–412
 - consequences in, 412–413
 - implementation of, 413
 - patterns related to, 413–414
 - problem addressed by, 410
 - sample model of, 414–415, 414f
 - structure of, 410, 411f
- protocol
 - for applications, 354
 - redundancy in, 355
 - and reliability, 355
- Protocol*, in ROOM Pattern, 195, 196
- protocol architecture, purpose of, 354–355
- protocol stack, in ISO communications
 - model, 354
- prototypes
 - focus of, 111
 - in iterative development, 107–108, 108f
 - mission of, 107, 125
 - requirements analysis for, 119
 - scheduling, 108
 - in semispiral lifecycle, 116
 - in spiral lifecycle, 110–111
 - testing, 125
 - use cases for, selection of, 119
- Proxy Pattern, 387–395
 - abstract of, 387
 - application of, 387
 - collaboration roles in, 388–391
 - consequences in, 391–392
 - implementation of, 392
 - patterns related to, 392–393
 - problem addressed by, 387–388
 - sample model of, 393–395, 393f, 394f
 - structure of, 388, 389f

pseudostates. *See also specific pseudostates*
 definition of, 38, 39f
 notation for, 470
 vs. states, 38
 Publish-Subscribe Pattern. *See* Observer
 Pattern

Q

qualities of service (QoS), 127–128
 in architecture evaluation, 118
 conflicts among, 56–57, 128
 control interfaces for, 170–171
 design and, 56
 in design patterns, 59
 in Distribution View, 75
 protocols and, 77
 for resources, 302
 qualities of service requirements
 definition of, 47
 in design patterns, 50–51
 Queue container, in Message Queuing
 Pattern, 210–211

R

radar channels, redundancy in, 79
 random faults
 definition of, 406
 in Homogenous Redundancy Pattern,
 415, 418–419
 in Triple Modular Redundancy Pattern,
 424
 Ready Queue
 in Dynamic Priority Pattern, 254
 in Static Priority Pattern, 246
 Real Machine, in Virtual Machine Pattern,
 179
 «realize» stereotype, 21
 real-time and embedded (RTE) systems
 hard, performance in, 75–76
 requirements of, xix
 soft, performance in, 76–77
 Real-Time UML Profile (UML Profile for
 Schedulability, Performance and Time),
 84–85
 Receiver, in Shared Memory Pattern, 359

recurrence properties strategy, for thread
 reification, 74
 Recursive Containment Pattern, 163–169
 abstract of, 164
 application of, 163
 collaboration roles in, 164
 consequences in, 165
 implementation of, 165–166
 patterns related to, 166
 problem addressed by, 164
 sample model of, 167–169, 167f, 168f,
 169f, 170f
 structure of, 164, 165f
 redundancy
 with Channel Architecture Pattern,
 157–158
 cost of, 409–410, 429–430
 and faults, 408
 implementation of, 93
 in protocols, 355
 in Safety and Reliability View, 79–81, 80f,
 406
 refactoring, 118
 «refine» stereotype, 21
 related information strategy, for thread
 reification, 73–74
 Relay Port, in ROOM Pattern, 196
 reliability. *See also* Safety and Reliability
 View
 definition of, 79, 406
 in Distribution View, 78–79
 fail-safe state in, 407
 protocols and, 355
 in Remote Method Calls, 366
 vs. safety, 140, 406–408, 407f
 Remote Method Call Pattern, 362–370
 abstract of, 362–363
 application of, 362
 collaboration roles in, 364–366
 consequences in, 366
 implementation of, 366–367
 patterns related to, 367–368
 problem addressed by, 363
 sample model of, 368–370, 368f, 369f
 structure of, 363, 364f
 Remote Notification Handle class
 in Broker Pattern, 399–400

- in Proxy Pattern, 391
- Remote Procedure Calls (RPCs), for
 - services, 362
- rendezvous
 - asynchronous, 92, 208–209
 - definition of, 72
 - forcing, 221–222
- Rendezvous* object, in Rendezvous Pattern, 229
- Rendezvous Pattern, 227–232
 - abstract of, 228
 - application of, 227
 - collaboration roles in, 229–230
 - consequences in, 230
 - implementation of, 230–231, 230f
 - patterns related to, 231
 - problem addressed by, 228
 - sample model for, 231–232, 231f–232f
 - structure of, 228f, 229
- requirements. *See also* functional
 - requirements; qualities of service requirements
 - capturing, 49
 - detailing, 120
 - negative, 119
 - prototype focus on, 111
 - specifying, 121
 - for subsystems, in System Engineering phase, 120
 - testing based on, 106–107, 108
 - in use cases, 47
- Requirements Analysis phase
 - in microcycle, 119–120
 - in semispiral lifecycle, 114, 115f, 119
 - validation test plan in, 125
- resource
 - definition of, 72, 302
 - implementation of, 92
 - management of, 302–308
- resource architecture. *See* Concurrency and Resource View
- Resource Client*
 - in Ordered Locking Pattern, 347
 - in Simultaneous Locking Pattern, 340–341
- Resource ID*, in Ordered Locking Pattern, 347
- Resource List*, in Ordered Locking Pattern, 347
- resource objects, semaphores for, 92
- Resource Pool Manager*, in Pool Allocation Pattern, 268
- ResourceMaster*, in Simultaneous Locking Pattern, 341
- restart, in fault handling, 408–409
- ReturnAction, 32
- reuse
 - of concepts, 126
 - of core services, 152
 - of virtual machine, 180–182
- reuse plan, defining, 117
- Rhapsody, 4–5
- risk
 - definition of, 80
 - in prototypes, 107
- role names
 - in composition, 17
 - definition of, 13
 - notation for, 13, 14f
- ROOM (Real-Time Object-Oriented Methodology) Pattern, 194–200
 - abstract of, 194–195
 - application of, 194
 - collaboration roles in, 196
 - consequences in, 196–197
 - implementation of, 197–198
 - patterns related to, 198
 - problem addressed by, 195
 - sample model of, 198–199, 199f, 200f
 - structure of, 195f
- ROPES (Rapid Object-Oriented Process for Embedded Systems). *See also specific phases*
 - architectural views in, 67, 68f
 - architecture in, 52, 57, 57f, 60, 67
 - diagrams delineated in, 25
 - flexibility of, 100–101
 - frameworks in, 109
 - iterative development in, 107, 109–110
 - logical architecture in, 60
 - macrocycle in (*See* macrocycle)
 - microcycle in (*See* microcycle)
 - mission of, 100–101
 - model execution in (*See* model execution)

- ROPES (*cont.*)
 - model-code associativity in, 105–106
 - nanocycle in (*See* nanocycle)
 - packages in, 26
 - physical architecture in, 67
 - prototypes in, 107–108, 108*f*
 - scalability of, 100, 101
 - testing in, 106–108, 109–110
 - use case detailing in, 49, 119–120
 - visual modeling in, 101
- Round Robin Pattern, 237–242
 - abstract of, 237
 - application of, 237
 - collaboration roles in, 239
 - consequences in, 240
 - implementation of, 240
 - patterns related to, 240
 - problem addressed by, 237
 - sample model of, 241–242, 241*f*
 - structure of, 238, 238*f*
- RPCs (Remote Procedure Calls), for services, 362
- RTE systems. *See* real-time and embedded systems
- S**
 - safety
 - definition of, 406
 - fail-safe state in, 407
 - vs.* reliability, 140, 406–408, 407*f*
 - risk in, 80
 - and software, 408
 - Safety and Reliability View
 - in architectural design, 123, 140
 - implementation of, 93
 - purpose of, 79, 140
 - redundancy in, 79–81, 80*f*
 - risk in, 80
 - Safety Coordinator* class, in Safety Executive Pattern, 453
 - Safety Executive* component, in Safety Executive Pattern, 453
 - Safety Executive Pattern, 450–456
 - abstract of, 450
 - application of, 450
 - collaboration roles in, 452–454
 - consequences in, 454
 - implementation of, 454
 - patterns related to, 454
 - problem addressed by, 451
 - sample model of, 455–456, 455*f*
 - structure of, 451–452, 451*f*
 - safety level strategy, for thread reification, 74
 - Safety Measure* class, in Safety Executive Pattern, 453
 - Safety Policy* class, in Safety Executive Pattern, 453
 - Sanity Check Channel*, in Sanity Check Pattern, 441
 - Sanity Check Pattern, 438–442
 - abstract of, 438–439
 - application of, 438
 - collaboration roles in, 439–441
 - consequences in, 441
 - implementation of, 442
 - patterns related to, 442
 - problem addressed by, 439
 - sample model for, 442, 443*f*
 - structure of, 439, 439*f*
 - scalability, 100
 - scaling, of architecture, 118
 - scenario modeling, of use cases, 49, 119
 - Scheduler*
 - in Critical Section Pattern, 310
 - in Cyclic Executive Pattern, 234
 - in Dynamic Priority Pattern, 254–255
 - in Highest Locker Pattern, 325–326
 - locking, 308–309
 - in Priority Ceiling Pattern, 333–334
 - in Priority Inheritance Pattern, 316
 - in Round Robin Pattern, 239
 - in Static Priority Pattern, 246
 - scheduling, in planning, in Party phase, 117
 - scheduling, of tasks
 - non-priority-based, 237–242
 - priority-based (*See* priority scheduling)
 - by virtual machines, 183
 - scope, 116–117
 - SCSI bus protocol, 355
 - throughput of, 77–78
 - Secondary Actuation Channel*, 418, 429

- Secondary Actuation Validation* component, 427–428
- Secondary Actuator* actor, 418, 428–429
- Secondary Data Transformation* component, 429
- Secondary Data Validation* component, 429
- Secondary Input Processing* component, 429
- Secondary Input Sensor* actor, 418, 429
- Secondary Output Processing* component, 429
- Segment*, in Garbage Compactor Pattern, 295–296
- semantic(s), of actions, 33
- semantic models
 - components of, 5–6
 - and diagrams, 7
- semantic objects, *vs.* subsystems, 27–28
- semaphores, 92
- semispiral lifecycle, 114–116, 115f
- «send» stereotype, 24
- SendAction, 32
- Sender* object, in Shared Memory Pattern, 359
- Sensor Input Processing* component
 - in Monitor-Actuator Pattern, 435
 - in Sanity Check Pattern, 441
 - in Watchdog Pattern, 447
- sequence diagram
 - advanced, notation for, 464
 - for architectural views, 67
 - for behavior, 121, 122, 123, 124
 - for collaboration structure, 124
 - concurrency in, 44–47, 46f
 - definition of, 44, 45f
 - function of, 6
 - in model execution, 104
 - notation for, 464
 - for requirements detailing, 120
 - for scenario modeling, 49, 119
- Sequencer*, in Virtual Machine Pattern, 179
- sequential substates, notation for, 469
- Sequential Watchdog, 448
- serial channels, 411
- server(s)
 - data shared between, 377
 - properties of, hiding, 387–388
 - with unknown properties, 395–396
- Server*, in Remote Method Call Pattern, 365
- Server* object, in Guarded Call Pattern, 223
- Server Stub*, in Remote Method Call Pattern, 366
- Server Thread*, in Guarded Call Pattern, 223
- Server-side Proxy*
 - in Broker Pattern, 400
 - in Proxy Pattern, 391
- service(s)
 - core set of, 151–152, 155
 - synchronous invocation of, 362–363
- service components, 152
- Set Point Source*
 - in Monitor-Actuator Pattern, 435
 - in Sanity Check Pattern, 441
- shallow history pseudostate, 39f, 40
- Shared Memory*, in Shared Memory Pattern, 359
- Shared Memory Pattern, 356–361
 - abstract of, 356–357
 - application of, 356
 - collaboration roles in, 357–359
 - consequences in, 359–360
 - implementation of, 360
 - patterns related to, 360
 - problem addressed by, 357
 - sample model of, 361, 361f–362f
 - structure of, 357, 358f
- Shared Resource* object
 - in Critical Section Pattern, 310–311
 - in Dynamic Priority Pattern, 255
 - in Guarded Call Pattern, 223–224
 - in Highest Locker Pattern, 326
 - in Ordered Locking Pattern, 347–348
 - in Priority Ceiling Pattern, 334
 - in Priority Inheritance Pattern, 316
 - in Simultaneous Locking Pattern, 341
 - in Static Priority Pattern, 246
- SignalEvent, 34
- Simultaneous Locking Pattern, 338–345
 - abstract of, 338
 - application of, 338
 - collaboration roles in, 339–341
 - consequences in, 341–342
 - implementation of, 342–343
 - patterns related to, 343
 - problem addressed by, 338

- Simultaneous Locking Pattern (*cont.*)
 - sample model of, 343–345, 343f–344f
 - structure of, 339, 339f
- single channels, for safety and reliability, 410
- single-event groups strategy, for thread reification, 73
- Sized Heap*, in Fixed Size Buffer Plan, 276
- sizing, 31, 31f
- Smart Pointer*, in Smart Pointer Pattern, 281–282
- Smart Pointer Pattern, 278–285
 - abstract of, 279
 - application of, 278–279
 - collaboration roles in, 281–282
 - consequences in, 282–284
 - implementation of, 284
 - patterns related to, 284
 - problem addressed by, 279–280
 - sample model of, 284, 285f
 - structure of, 280, 281f
- soft real-time systems, performance in, 76–77
- software
 - complexity of, 97
 - development plan for, 116–117
 - and safety, 408
- Software Component Level, in physical architecture abstraction, 65f, 66
- software subsystems, purpose of, 69
- Solution, in design patterns, 51, 59, 130
- source level code
 - for class diagrams, 10–12
 - generation of, 6, 124–125
 - legacy, integration of, 125
 - in ROPES process, 105–106
 - testing, 125
 - and visual model, association of, 105–106
- source-level languages, binding, in CORBA, 83–84
- specialization
 - definition of, 18
 - notation for, 461
- specification
 - for requirements capture, 49
 - for use case detailing, 119
- spiral development lifecycle. *See also* ROPES
 - drawbacks in using, 113–114
 - testing in, 110
 - vs.* waterfall lifecycle, 110
- stability, 242
- Stack*
 - in Dynamic Priority Pattern, 255
 - in Round Robin Pattern, 239
 - in Static Priority Pattern, 246
- standard(s), benefits of using, 4–5
- Standard Template Library, containers in, 90
- state(s)
 - nested, notation for, 468
 - vs.* pseudostates, 38
- state icon, notation for, 468
- state machines
 - finite (FSM), 34
 - statecharts for, 124
- statecharts
 - for architectural views, 67
 - for behavior, 122, 124
 - benefits of, 34
 - for control interfaces, 170–171
 - definition of, 8
 - elements of, 34, 35f
 - function of, 6
 - in Hierarchical Control Pattern, 174
 - in model execution, 104
 - notation for, 468–471
 - for requirements detailing, 120
 - in ROOM Pattern, 194–195
 - for specification, 49, 119
 - for state machines, 124
 - in System Engineering phase, 121
- Static Allocation Pattern, 260–265
 - abstract of, 260
 - application of, 260
 - collaboration roles in, 263
 - consequences in, 263–264
 - implementation of, 264
 - patterns related to, 264
 - problem addressed by, 260–262
 - sample model for, 265, 265f
 - structure of, 262, 262f
- static priority, 242

- Static Priority Pattern, 242–251
 - abstract of, 242–243
 - application of, 242
 - collaboration rolls in, 245–247
 - consequences in, 247–248
 - implementation of, 248
 - patterns related to, 248–249
 - problem addressed by, 243
 - sample model of, 249–251, 250f
 - structure of, 243–244, 244f
- stereotypes
 - definition of, 9
 - of dependencies, 21
 - function of, 17
 - of nodes, 81
 - notation for, 14f, 17, 81, 82f, 462
- strategic defects, expense of, 106
- struct(s), for class implementation, 86–87
- structural aspect, of models, 6
- structural design patterns, 128
- structural diagrams, 24–25, 67
- structure
 - class diagrams for, 123
 - in UML, 84
- stub pseudostate, 39f, 41
- subactivities, in processes, 98
- subclasses
 - methods in, 19–20
 - for specialization, 18
- submachines, notation for, 471
- substates
 - concurrency with, 37
 - notation for, 470
 - vs.* *or-states*, 36, 37, 37f
 - sequential, notation for, 469
 - synchronization of, 37–38
- substitutability, 21
- subsystem(s)
 - aspects of, 28
 - classes in, 139
 - and components, 28, 65f, 70–71, 139, 185
 - content of, 138–139
 - core services and, 152
 - criteria for inclusion in, 69
 - definition of, 6, 70, 138
 - in deployment models, 30
 - domains in, 138
 - function of, 91
 - implementation of, 91–92
 - instances in, 139
 - notation for, 28, 29f
 - packages and, 27
 - semantic objects, 27–28
 - sizing, 31, 31f
 - software, 69
 - in Systems Engineering phase, 59
 - threads in, 139
 - use cases for, 49
 - use of, 27–28, 70
- Subsystem and Component View
 - in architectural design, 123, 138
 - component diagram for, 71, 71f
 - implementation of, 91–92
 - purpose of, 52, 68
 - subsystem diagram for, 69, 70f
 - in System Engineering phase, , 120
- subsystem architecture. *See* Subsystem and Component View
- subsystem diagram
 - components in, 123
 - for Subsystem and Component View, 69, 70f
 - in System Engineering phase, 121
- subsystem interfaces, in System Engineering phase, 120
- «subsystem» stereotype, 28, 29f
- superclasses
 - extension of, 20–21
 - generalization for, 18
 - methods in, 19–20
 - specialization of (*See* subclasses)
- Switch To Backup Pattern. *See* Homogeneous Redundancy Pattern
- symmetric architecture
 - definition of, 83
 - deployment diagrams for, 81
 - distribution implementation for, 93
 - objects in, 75
- symmetric distribution architecture, 354
- Synch Policy, in Rendezvous Pattern, 229–230
- synch pseudostates
 - definition of, 39, 39f
 - notation for, 471

- synchronous event transfer, implementation of, 36
 - system(s)
 - in scenario modeling, 49
 - sizing, 31, 31f
 - system development, model execution *vs.* traditional, 103
 - system functional aspect, of models, 6
 - System Level, in physical architecture abstraction, 65, 65f
 - system model, architectural aspects in, 52
 - System Object*, in Static Allocation Pattern, 263
 - System View, 68, 69f
 - systematic faults
 - definition of, 406
 - in Heterogeneous Redundancy Pattern, 426–427
 - in Homogenous Redundancy Pattern, 415, 419
 - in Triple Modular Redundancy Pattern, 424
 - Systems Engineering Level, in physical architecture abstraction, 65–66, 65f
 - Systems Engineering phase
 - activities in, 59, 120
 - design in, 58–59
 - integration test plan in, 125
 - in microcycle, 120–121
 - purpose of, 59, 120
 - in semispiral lifecycle, 114–115, 115f
 - Subsystem View in, 69
- T**
- Target*, in Smart Pointer Pattern, 282
 - target object strategy, for thread reification, 74
 - Target Wrapper*, in Smart Pointer Pattern, 282
 - task(s)
 - blocking, 303–305, 304f
 - priority scheduling for, 303, 308–309
 - Task Control Block (TCB)*
 - in Critical Section Pattern, 311
 - in Dynamic Priority Pattern, 255
 - in Highest Locker Pattern, 326
 - in Priority Ceiling Pattern, 334
 - in Priority Inheritance Pattern, 317
 - in Round Robin Pattern, 239
 - in Static Priority Pattern, 246
 - task diagram, for Concurrency and Resource View, 72, 73f
 - task rendezvous, in Deployment View implementation, 93
 - «Task» stereotype, for Concurrency and Resource View, 72, 73f
 - TCP/IP
 - and QoS requirements, 77
 - reliability of, 355
 - TDMA (Time Division Multiple Access), 77, 355
 - technical disciplines, of subsystems, in System Engineering phase, 120
 - technology, prototype focus on, 111
 - terminal pseudostate, 38–39, 39f
 - TerminateAction, 32
 - Test phase, 112f, 113, 125–126
 - testing
 - automating, 108
 - of patterns, 133
 - of prototypes, against mission, 125
 - requirements-based, 106–107, 108
 - in ROPES process, 106–108, 109–110
 - of source code, 125
 - in spiral lifecycle, 110
 - in waterfall lifecycle, 110
 - textual descriptions
 - for interfaces, 121
 - for requirements detailing, 120
 - this* pointer, for member functions, 87
 - thread(s)
 - «active» object management of, 139, 204–206, 205f
 - communication between, 207–208, 227–228
 - in concurrency, 204–206, 205f
 - creation of, 92
 - destruction of, 92
 - properties of, 206
 - reification of, strategies for, 72–74
 - in subsystems, 139
 - synchronization of (*See* rendezvous)
 - Thread Level, in physical architecture abstraction, 65f, 66

- Thread* object, in Message Queuing Pattern, 210
- throughput
 - architecture for (*See* Channel Architecture Pattern)
 - in Distribution View, 77–78
- time, in sequence diagram, 44
- Time Division Multiple Access (TDMA), 77
- time event, 36
- Timebase*
 - in Safety Executive Pattern, 453
 - in Watchdog Pattern, 447
- TimeEvent, 34
- Timer*, in Round Robin Pattern, 239
- timing constraints, in sequence diagram, 44, 45f
- tools, for UML, 4–5
- training, for UML, 5
- transient faults
 - in Monitor-Actuator Pattern, 436
 - in Protected Single Channel Pattern, 413
- transition(s)
 - notation for, 468
 - in statecharts, 34
- transition actions, guards and, 36
- Translation phase, 112f, 113, 124–125
- translation tools, for application creation, 85
- Transport Protocol Domain*, in 5-Layer Architecture Pattern, 149–150
- Triple Modular Redundancy Pattern, 421–426
 - abstract of, 421
 - application of, 421
 - collaboration roles in, 423
 - consequences in, 424
 - faults in, 424
 - implementation of, 424
 - patterns related to, 424–425
 - problem addressed by, 422
 - sample model of, 425–426, 425f
 - structure of, 422–423, 422f
- vs.* CORBA, 84
- definition of, 4
- diagrams in, 4, 7, 24, 25 (*See also specific diagrams*)
- notation in, 4, 458–472
- tools for, 4–5
- training for, 5
- versatility of, 5
- UML metamodel
 - components of, 32
 - purpose of, 4
- UML Profile for Schedulability, Performance and Time (Real-Time UML Profile)*, 84–85
- UML-RT. *See* ROOM Pattern
- unbounded priority inversion, 305, 306f, 314, 323
- Unified Modeling Language. *See* UML
- uninitialized pointers, 280
- UninterpretedAction, 32
- Unit Type*, in Data Bus Pattern, 382
- urgency, 242
- Usage dependency, 24
- use case(s)
 - for architectural views, 67
 - for behavior, 121
 - contents of, 121
 - definition of, 47
 - detailing, 49, 119–120
 - function of, 6
 - notation for, 47–48, 48f, 465
 - for prototypes, 111, 119
 - scenario modeling of, 49
 - selection of, 117
 - for subsystems, 120
 - in Subsystems and Components View, 91
- use case diagram, notation for, 465
- use case relationships, notation for, 465
- User Interface Domain*, in Five-Layer Architecture Pattern, 149

U

- UDP, reliability of, 355
- UML (Unified Modeling Language)

V

- Validation testing, of prototypes, 125
- “Victorian novel” approach, for specification, 49

views
 in application model, 6–7
 diagrams for, 25
virtual function table (VTBL), for
 generalization in C++, 89
Virtual Machine component, 179,
 180–184
Virtual Machine Pattern, 176–184
 abstract of, 176–177
 application of, 176
 collaboration roles in, 178–179
 consequences in, 179–180
 implementation of, 180–184
 patterns related to, 184
 problem addressed by, 177
 structure of, 177^f
virtual methods, 89
visibility, notation for, 458
visual debugger, in model execution,
 103
visual modeling, in ROPES process, 101,
 102^f

W*Watchdog*

 in Safety Executive Pattern, 453
 in Watchdog Pattern, 447
watchdog, definition of, 444
Watchdog Pattern, 443–450
 abstract of, 444
 application of, 443–444
 collaboration roles in, 445–447
 consequences in, 447–448
 implementation of, 448–449
 patterns related to, 449
 problem addressed by, 444
 sample model of, 449–450, 449^f
 structure of, 444–445, 445^f, 446^f
waterfall lifecycle, 110, 115–116
worker roles, in processes, 98, 99^f

X

XP (Extreme Programming) approach, in
 nanocycle, 112