

Index

A

- Abstract classes, 206, 655
 - reuse categories, 216–217
 - software product lines, 207–211
- Abstract data type, 20, 655
- Abstract interface, 23, 655
- Abstract kernel classes, 216, 408, 478, 572
- Abstract operation, 206, 655
- Abstract optional classes, 217
- Abstract use cases, 75, 84–85, 88–89
- Actions, 172, 620, 655
 - feature-dependent, 189
 - on statecharts, 174
- Active classes, 630
- Active objects, 32, 622, 630, 655
- Activities, 174, 620, 655
 - feature-dependent, 189
- Actors, 12–13, 45, 47, 66, 614, 655
 - product line use cases, 73
- Agent objects, 139
- Agent-Based Negotiation pattern, 266
- Agents, 246–247, 266–269
- Aggregate classes, 27, 617, 655
- Aggregate subsystems, 293, 655
- Algorithm objects, 139, 655
- Alternative communication diagrams, 152–155
- Alternative features, 7, 97, 217, 656
 - in microwave oven product line, 357, 359–360
- Alternative functionality, 76–77
- Alternative message sequences, 158
- Alternative requirements, 65
- Alternative use cases, 47–48, 50, 65, 69–71, 83, 656
 - in electronic commerce product line, 70–71
- Analysis modeling, 12–13, 54, 56, 656
- Analysis patterns, 11
- Application analysis modeling, 322
- Application design modeling, 322
- Application deployment, 656
- Application domain, 5
- Application engineering, 6, 319–344, 656
- Application logic objects, 59, 139, 656
- Architectural communication patterns,
 - in electronic commerce product line, 486–487
 - in factory automation product line, 276, 595–599
 - in microwave oven product line, 421, 424
- Architectural patterns, 11, 232, 656
- Architectural styles, 232
- Architecture and design reuse, 5
- Architecture diagrams, 339
- Architectures, configurable, 281
- Associations, 108, 656
 - multiplicity, 26, 616
 - on product line context class diagram, 126

688 Index

- Asynchronous (loosely coupled) communication, 35–36
 - Asynchronous message communication, 32, 253, 288, 656
 - Asynchronous Message Communication pattern, 253–254, 283, 339, 421, 487, 596, 640
 - Asynchronous Message Communication with Callback pattern, 254–257, 303, 641
 - Asynchronous messages, 253–254, 257, 623
 - At-least-one-of feature groups, 113–115, 656
 - Atomic transactions, 270–272
 - Attributes, 21, 615
- B**
- Base use case, 78–79, 84
 - Behavioral analysis, 656
 - Behavioral models, 656
 - Bidirectional Asynchronous Message Communication pattern, 254–255, 596, 642
 - Bidirectional asynchronous messages, 254–255
 - Binary association, 616
 - Binary semaphore, 656
 - Black box specification, 656
 - Black box testing, 49, 323
 - Boolean guard condition, 188–189, 399
 - Boundary objects, 138, 656
 - Broadcast communication, 264, 657
 - Broadcast pattern, 264, 643
 - Broker architectural pattern, 245–246, 632
 - Broker communication patterns, 260–262
 - Broker Forwarding pattern, 260, 644
 - Broker Handle pattern, 260–262, 424, 487, 599, 645
 - Brokering services, 260
 - Brokers, 245–246, 260–262, 657
 - Business logic objects, 139, 657
- C**
- Callback, 303, 314, 657
 - Centralized Control architectural pattern, 247–248, 250–251, 418, 633
 - Child classes, 28, 30
 - Child state machines, 181–182
 - Class diagrams, 616, 657
 - aggregations, 27, 617
 - associations, 26, 616
 - composition hierarchies, 27, 617
 - feature-based, 225–227,
 - generalization/specialization hierarchy, 28, 617
 - visibility, 617–618
 - Class interface specifications, 657
 - Class structuring criteria, 139, 657
 - Classes, 11, 20–21, 615, 657
 - aggregation hierarchies, 27–28
 - associations, 26
 - attributes, 21, 30
 - binary association, 616
 - categories, 47, 120–121
 - composite, 122
 - entity, 121
 - external, 120
 - generalization/specialization, 28, 30
 - inheritance, 28, 30
 - instances of, 20–21
 - many-to-many association, 26
 - numerically specified association, 26
 - one-to-many association, 26
 - one-to-one association, 26
 - optional association, 26
 - physical, 121
 - relationships between, 26–30
 - relationships with features, 205
 - reuse categorization, 213–217
 - role stereotypes, 213
 - state-dependent, 169
 - structured, 307
 - structuring criteria, 139
 - subclasses, 28, 30
 - variant, 122
 - variation points, 206–213
 - whole/part relationship, 27
 - Client/Agent/Server architectural pattern, 246–247, 480, 634
 - Client/Server architectural pattern, 240–245, 635
 - factory automation product line case study, 590–591
 - Collaboration diagram, 141, 618, 657
 - COM (Component Object Model), 36–37, 657, 658
 - COMET (Concurrent Object Modeling and Architectural Design Method), 12–13, 43, 657, 659
 - Committed transactions, 270

- Common features, 7, 97–98, 217, 657
- Common requirements, 96–98
- Commonality, 6, 120–122, 657
- Commonality/variability analysis, 89, 97–101, 357, 657
- Communicating Components architectural pattern, 251–253
- Communication
 - between components, 282
 - multicast, 264–266
 - negotiated patterns, 266–269
- Communication diagrams, 47, 142, 284, 614, 618–619, 658
 - feature-based, 223–225, 581
 - generic form, 146
 - instance form, 146
 - integrated, 284–285
 - versus* sequence diagrams, 145–146
- Communication models, integrating, 284–285
- Complex ports, 308, 658
- Component instances, 325–326
- Component interfaces, 306–311
 - designing, 306–311
 - inheritance design, 314, 316
- Component ports, 308
- Component structuring criteria, 294–300, 658
- Component technologies, 37
- Component-based distributed design phase, 279
- Component-based software architecture, 45
 - design, 285–289
 - designing distributed components, 288
 - in electronic commerce product line, 487, 490, 494
 - in microwave oven product line, 418, 420–421
- Component-based software design, 279
- Component-based systems, 34, 279, 658
 - components, 35, 281
 - connectors, 35–36, 306
 - discovery services, 39–40
 - distributed component communication protocols, 37–38
 - middleware, 36–37
 - registration services, 38–39
 - wrapper components, 40–41
- Components, 34, 279, 658
 - composite, 281
 - concurrent, 311
 - concurrent server, 302–305
 - configurable, 281
 - connectors, 35–36, 306
 - control, 297
 - coordinator, 299
 - coupling between, 280
 - data analysis, 300
 - data collection, 299–300
 - definition of, 35
 - designing, 311–316, 487, 490, 494
 - distributed, 281, 288, 311
 - interfaces, 307–311
 - localized autonomy, 292
 - mapping to different nodes, 285
 - message communication, 282–283
 - multiple interfaces, 307
 - operations provided by, 35
 - parameterized, 9
 - passive, 311
 - peer, 288
 - performance, 292
 - plug-compatible, 311–313
 - ports, 308–309
 - proximity to source of physical data, 292
 - required operations, 35
 - scope of control, 293
 - self-contained, 281
 - sequential server, 301–302
 - servers, 291
 - simple, 281, 311
 - specialized hardware, 292–294
 - structure of, 282
 - structured classes, 307
 - synchronous message communication, 36, 259
 - UML 2.0 modeling, 307
 - unidirectional or bidirectional communication, 283
 - variable, 313–314
- Composite classes, 122, 290–291
- Composite components, 281, 311, 435, 437–438, 658
- Composite objects, 290–291, 658
- Composite states, 176, 658
- Composite structure diagrams, 307, 614, 658
- Composite subsystems, 293, 658

690 Index

- Composition, 27–28, 290, 658
 - Composition hierarchies, 27, 617
 - Composition relationship, 27
 - Compound Transaction pattern, 272–273, 651
 - Compound transactions, 272–273
 - Concrete classes, 217, 658
 - Concrete kernel class, 217
 - Concrete optional class, 217
 - Concrete use case, 84
 - Concrete variant class, 217
 - Concurrent application, 32
 - Concurrent communication diagrams, 253, 622–624, 659
 - Concurrent components, 251–253, 311, 435, 438
 - Concurrent message sequences, 157–158, 164
 - Concurrent objects, 32–34, 622, 659
 - Concurrent process, 659
 - Concurrent server, 302, 659
 - Concurrent server components, 302–304
 - Concurrent server subsystems, 305
 - Concurrent tasks, 32, 659
 - Conditions, 172, 620, 659
 - Configurable architectures, 281
 - Configurable components, 281
 - Configuration parameters, 8
 - Connectors, 35–36, 306, 659
 - Constraints, 189, 627–628, 659
 - Construction phase, 55, 57
 - Context models, 127–130
 - Control classes
 - state-dependent, 180
 - Control components, 250, 297, 420, 588
 - events, 247–248
 - interfaces, 434
 - Control objects, 59, 138–139, 159, 659
 - Control patterns, implications of, 250–251
 - Coordinator components, 299, 588
 - Coordinator objects, 138, 659
 - CORBA (Common Object Request Broker Architecture), 36–37, 39, 657, 659
- D**
- Data, distribution of, 305–306
 - Data abstraction, 23, 659
 - Data abstraction classes, 138, 659
 - Data analysis components, 300
 - Data collection components, 299–300
 - Data replication, 306, 660
 - Data structure, 23–24
 - Database wrapper classes, 138, 659
 - DCE (Distributed Computing Environment), 36
 - Default class, 215–216, 660
 - Default feature, 99, 660
 - Default objects, 159
 - Default parameterized classes, 216
 - Delegation connector, 311, 660
 - Deployment diagram, 326, 614, 625–626, 660
 - Derived classes, 28
 - Design
 - for change, 23
 - concept, 660
 - modeling variability, 8–10
 - Design method, 12, 14, 660
 - Design modeling, 12–13, 54, 59–60, 630, 660
 - component-based software architecture, 45
 - in electronic commerce product line, 480–500
 - in factory automation product line, 587–609
 - in microwave oven product line, 418–438
 - Design notation, 11, 660
 - Design patterns, 10–11, 232, 660
 - Device interface object, 138, 660
 - Discovery pattern, 262–263, 646
 - Discovery services, 39–40
 - Distributed Agents pattern, 246
 - Distributed applications, 281, 660
 - concurrent message-based design, 281
 - constituent components, 282
 - deploying, 282
 - design steps, 281–282
 - group communication, 263–266
 - passive data abstraction object, 301
 - software architecture, 282–285
 - synchronizing updates on different nodes, 270–271
 - Distributed communication, 259
 - Distributed component communication protocols, 37–38
 - Distributed components, 281, 311, 660
 - designing, 288
 - technologies, 37
 - Distributed Control architectural pattern, 314, 636
 - in factory automation product line, 248, 250–251, 591

- in high-volume manufacturing system case study, 248
 - Distributed servers, 306, 661
 - Distributed systems, 270–271
 - Distribution of data, 305–306
 - Documenting
 - product line use cases, 72–73
 - small variations in use cases, 75–76
 - software architectural patterns, 275
 - Domain analysis, 6, 661
 - Domain modeling, 6, 661
 - Domain-specific pattern, 661
 - Domain-specific software architectures, 6, 661
 - Double spiral model, 52–53
 - DSSA (Domain-Specific Software Architecture), 9
 - Dynamic analysis, 661
 - communicating state-dependent objects, 199–204
 - kernel first approach, 148–150
 - Dynamic interaction between objects, 141
 - Dynamic interaction modeling, 170
 - Dynamic modeling, 12–13, 31, 47, 59, 141, 661
 - communication diagrams, 50, 142, 322
 - in electronic commerce product line, 464–466
 - evolutionary, 147–148
 - in factory automation product line, 535–541
 - features, 322
 - message sequence description, 142
 - in microwave oven product line, 367–379
 - object interaction modeling, 142–144
 - object reuse categorization, 150
 - optional and alternative communication diagrams, 50
 - sequence diagrams, 144–145
 - single systems, 142–147
 - use cases, 322
- E**
- EDLC (Evolutionary Domain Life Cycle), 7–10
 - EJB (Enterprise JavaBeans), 37, 661
 - Elaboration phase, 55–57
 - Embedded system product lines, 122
 - Encapsulation, 23, 661
 - Entity class modeling, 50, 131–133, 135
 - Entity classes, 121, 661
 - Entity objects, 59, 138, 661
 - Entity relationship modeling, 131
 - Entry actions, 174, 620
 - ESPLEP (Evolutionary Software Product Line Engineering Process), 43–45, 661
 - Event notification, 265, 305
 - Event subscription, 265, 305
 - Event synchronization, 33–34, 239, 662
 - Events, 171–172, 247–248, 620, 661
 - Evolutionary dynamic analysis, 147–148
 - analyzing impact of variation points and features, 164–168
 - kernel first approach, 158–159
 - in microwave oven product line, 158–168
 - product line evolution approach, 161
 - Evolutionary dynamic modeling, 147–148
 - Evolutionary product line development, 50
 - Evolutionary Software Product Line Engineering Process (ESPLEP), 43–45, 662
 - Exactly-one-of feature groups, 110, 112–113, 115, 219, 662
 - Exit actions, 174, 620, 662
 - Explicit features, 101, 108, 662
 - Extend relationships, 75, 77–84, 614
 - Extension conditions, 79–80
 - Extension points, 79, 83–84
 - Extension use cases, 78–81, 516
 - External classes, 120, 127, 662
 - External devices, 124
 - External input device, 124–125, 127
 - External input/output device, 124–125
 - External interface, 23
 - External I/O device actors, 67
 - External objects, interfacing to, 293
 - External output device, 124–125
 - External systems and actors, 68–69
- F**
- Family of systems, 5–6, 662
 - FAST (Family-Oriented Abstraction, Specification, and Translation), 6, 9
 - Feature analysis, 56, 95
 - analyzing optional and alternative features, 97
 - commonality/variability, 97–101
 - complexity, 96
 - goal of, 102

692 Index

- Feature condition, 188–189, 399
- Feature dependency diagram, 107–108, 362, 523
- Feature groups, 110–115, 662
 - aggregation, 112–113
 - at-least-one-of, 113–115
 - exactly-one-of, 112–113, 115, 219
 - in microwave oven product line, 359–360
 - mutually exclusive, 111–112
 - stereotypes, 110
 - tables, 114–115
 - tagged values, 110
 - zero-or-more-of, 114, 116
 - zero-or-one-of, 111–112
- Feature modeling, 7–8, 47, 58, 95, 662
 - commonality/variability analysis, 357
 - in electronic commerce product line, 447–451
 - in factory automation product line, 107, 516–525
 - in microwave oven product line, 357–364
 - with UML, 107–110, 115–118
 - as use case packages, 107
- Feature packages, 223
- Feature-based class diagrams, 225–227, 584
- Feature-based communication diagrams, 223–225, 581, 584
- Feature-based impact analysis, 161, 219–223, 662
- Feature/class dependencies, 48, 217–219, 611, 662
- Feature/class dependency analysis, 59, 225, 662
 - in electronic commerce product line, 477–480
 - in factory automation product line, 572–587
 - in microwave oven product line, 408–418
- Feature/class dependency modeling, 205, 223
- Feature/class dependency tables, 227–230
- Feature-dependent actions, 189
- Feature-dependent activities, 189
- Feature-dependent states, 189
- Feature/object dependencies, 223–230, 575
- Features, 7, 56, 58, 95–96, 523, 662
 - alternative, 7, 97, 217, 357, 525
 - analyzing impact of, 164–168
 - Boolean guard condition, 399
 - common, 7, 97, 217
 - constraints, 108
 - default, 99
 - definition of, 47
 - dependencies, 96, 100, 108, 227, 362
 - explicit, 108
 - feature condition, 188–189, 399
 - feature modeling, 205
 - functional, 7, 106
 - grouping use cases into, 449
 - impact of, 161
 - implicit, 108
 - interdependent, 103–104
 - mandatory, 7
 - mutually exclusive, 7, 97, 99, 111–112
 - mutually includes dependency, 525
 - mutually inclusive relationships, 362
 - nonfunctional, 7
 - objects required to support, 223–225
 - optional, 7, 97, 217, 357, 525
 - parameters, 7–9, 99–100
 - relationship
 - with classes, 205
 - with parameterized classes, 227–228
 - representing in tables, 108–110
 - requires dependency, 108
 - stereotypes, 107
 - use cases, 101–107
- Feature/use case dependencies, 101–107
- Feature/use case dependency table, 108–110, 320, 322
- Finite state machine modeling, 59, 169
- Finite state machines, 59, 169, 663
 - actions, 172, 174
 - entry actions, 174
 - events, 171–172
 - exit actions, 174
 - guard conditions, 172
 - kernel, 170–176
 - orthogonal, 179, 375
 - state transition, 170
 - state-dependent control objects, 199–200
 - state-dependent objects, 170–171
 - states, 171
- Flat statecharts, 169, 176
- Flat transactions, 272
- FODA (feature-oriented domain analysis), 7–9, 96
- Forward evolutionary engineering, 49–50, 663
- FTP (File Transfer Protocol), 38
- Functional components, 5

- Functional features, 7, 102–103, 106
 - Functional requirements, 65
 - Functional testing, 48–49, 323
- G**
- Generalization/specialization hierarchy, 28, 30, 617, 663
 - Generalized classes, 28, 30,
 - Generic architecture, 6
 - Generic communication diagram, 146, 224
 - Group message communication patterns, 263–266
 - Groupcast communication, 263–266
 - Guard conditions, 172, 189
- H**
- Hierarchical Control architectural pattern, 250–251, 314, 591, 637
 - Hierarchical Layers pattern, 232
 - Hierarchical state decomposition, 169
 - Hierarchical statecharts, 169
 - History state, 178
 - HTTP (HyperText Transfer Protocol), 38
- I**
- Identifying use cases, 71–72
 - Idioms, 11, 232, 663
 - Impact analysis, 219–223
 - Implicit features, 101, 663
 - Implicit mutually inclusive features, 108
 - Inception phase, 54–55
 - Include relationship, 75, 614
 - Inclusion use case, 84
 - Incremental application implementation, 322
 - Incremental component implementation, 48
 - Incremental development model, 51
 - Incremental prototype, 323
 - Incremental software development, 663
 - Information hiding, 6, 9, 20, 22–25, 290, 663
 - Information hiding class, 25, 663
 - Information hiding objects, 23, 663
 - Inheritance, 11, 20, 28, 30, 663
 - component interface design, 314, 316
 - modeling design variability, 9–10
 - Inheritance relationships, 28, 30, 136, 617
 - Inherited state machines, 169, 181–188, 193
 - Input components, 300, 420
 - Input device interface classes and objects, 138, 159
 - Input/output device interface object, 138
 - Integrated communication diagrams, 284–285, 663
 - Integration testing, 48, 323–324
 - Interaction diagrams, 146, 663
 - communication diagrams, 142, 618–619
 - generic form, 146
 - instance form, 146
 - sequence diagrams, 144, 619
 - Interdependent features, 103–104
 - Interface objects, 59, 138, 149, 664
 - Interfaces, 20, 23, 663
 - components, 307–311, 600, 603–604
 - definition of, 307
 - provided, 308–311
 - required, 308–311
 - Internal objects, 149
 - I/O components, 300
 - Iterative software development, 664
- J**
- J2EE (Java 2 Platform Enterprise Edition), 36, 38
 - Java RMI (remote method invocation), 36
 - JavaBeans, 37, 664
 - Jini, 37, 664
- K**
- Kernel architectural pattern, 239–240, 590, 638
 - Kernel classes, 59, 120, 213–214, 664
 - factory automation software product line, 572–576
 - microwave oven product line, 367–368, 411
 - Kernel communication diagrams, 148
 - Kernel components, 57, 477–478, 664
 - Kernel features, 98
 - Kernel first approach, 50, 148, 664
 - alternative sequences, 150
 - evolutionary dynamic analysis, 158–159
 - microwave oven product line, 369, 372–375
 - non-state-dependent dynamic analysis, 149–150
 - state-dependent dynamic analysis, 193–197
 - variation points, 150
 - Kernel interfaces, 307
 - Kernel objects, 158–159, 664
 - Kernel system, 127, 148–149, 664

694 Index

- Kernel use case model, 89
- Kernel use cases, 47–50, 65, 69–71, 664
- KobrA approach, 10

- L**
- Layered component-based architecture, 232, 588–594
- Layered software architecture, 232, 482, 484
- Layers of Abstraction architectural pattern, 232, 480, 587, 639
- Legacy applications, 40–41
- Legacy databases, 41, 456
- Location transparency, 260
- Long-Living Transaction pattern, 273–275, 652
- Long-living transactions, 273–275
- Loosely coupled message communication, 253, 664

- M**
- Managing variability, 60
- Mandatory alternative, 77
- Mandatory features, 7, 98
- Many-to-many association, 102
- Message labels on interaction diagrams, 156
- Message sequence description, 142, 664–665
- Message sequence number, 155–158
- Messages, 155, 253, 283–284, 624, 629–630, 664–665
 - alternative sequence, 166
 - argument list, 156
 - asynchronous, 253–254, 257, 623
 - bidirectional, 283
 - bidirectional asynchronous, 254–255
 - concurrent sequences, 164
 - conditional sequence, 164
 - return values, 156–157
 - synchronous, 255–257, 623
 - unidirectional, 283
- Metaclasses, 107–107, 110–111
- Microkernel pattern, 239
- Middleware, 36–37, 665
- Modeling
 - components with UML 2.0, 307
 - design variability, 8–10
 - inheritance, 9–10
 - parameterization, 8–9
 - feature dependencies as use case dependencies, 103–104
 - functional features as groups of use cases, 102–103
 - functional features with variation points, 106
 - optional functionality variation point, 76
 - parameterized features as variation points, 106–107
 - product line variability with extension points, 83–84
 - product line variability with features, 97
 - single systems, 11–13
 - small variations in use cases, 75–77
 - variability
 - with extend relationship, 77–84
 - with include relationship, 84–89
 - in use cases, 74–75
- Multi-Agent Negotiation pattern, 266
- Multicast communication, 264–266, 665
- Multilevel Control pattern, 250
- Multiple readers and writers, 33, 303–304, 665
- Multiple-Client/Multiple-Server pattern, 241–242, 245
- Multiple-Client/Single-Server pattern, 245
- Multiplicity of associations, 26, 616
- Mutual exclusion problem, 33, 239, 302, 665
- Mutually exclusive constraint, 108, 122
- Mutually exclusive feature groups, 219, 665
- Mutually exclusive features, 7, 111–112
- Mutually exclusive subclasses, 122, 364
- Mutually includes association, 108
- Mutually inclusive features, 101, 361

- N**
- Name service, 260
- Necessary features, 98
- Negotiable services, 266
- Negotiated communication, 266, 665
- Negotiated communication patterns, 266–269
- Negotiation pattern, 266–269, 647
- .NET, 37–38
- Nonfunctional features, 7
- Non-state-dependent dynamic analysis, 148–150
- Non-state-dependent kernel first approach, 150
- Numerically specified association, 26

O

Object and class structuring, 136–139, 158–159
Object Broker pattern, 245
Object brokers, 39, 665
Object diagrams, 614
Object interaction diagrams, 47
Object interaction modeling, 12, 142–144
Object request broker, 245, 665
Object Request Broker pattern, 245
Object structuring, 136–139, 535, 537
Object structuring criteria, 139, 665
Object-oriented analysis, 12, 665
Object-oriented concepts, 11, 19–22
Object-oriented design, 22–26, 665
Objects, 20–21, 615, 629, 665
 active, 32
 categories, 120–121
 composite, 290–291
 concurrent, 32
 dynamic interaction between, 31, 141
 encapsulation, 23
 information hiding, 23
 passive, 32
 structuring criteria, 139
OMG (Object Management Group), 15, 39
One-and-only-one-of feature group, 112, 666
One-to-many association, 26
One-to-one association, 26, 28
Operating system, 239–240
Operations, 20–22, 24, 666
Optional actors, 71
Optional alternative, 77
Optional association, 26
Optional classes, 59, 120, 215, 666
Optional communication diagrams, 152
Optional components, 4, 57, 425–426, 666
Optional external classes
Optional features, 7, 97, 217, 357, 525, 576–579, 666
 commonality/variability feature analysis, 98
 factory automation software product line, 517
 feature-based impact analysis, 220–222
 microwave oven product line, 359, 361
 prerequisite features, 100
 variation points, 106
Optional functionality, 76
Optional objects, 221, 408, 666

Optional parameterized classes, 216
Optional requirements, 65
Optional use cases, 47–48, 50, 56, 65, 69–71, 666
 communication diagrams, 152
 in factory automation software product line, 517
 impact on features, 167–168
 iteratively developed, 89
Optional variability, 83
ORB (Object Request Broker) middleware, 39
Orthogonal statecharts, 178–180, 547
OSI protocol, 236
Output components, 300, 420
Output device interface objects, 138

P

Packages, 102–103, 223, 622, 666
Parameterized classes, 211
 in microwave oven product line, 211–213
Parameterized components, 9, 435
Parameterized features, 99–100, 666
 in microwave oven product line, 361–362
 modeling as variation points, 106–107
Parameterized state machines, 169, 188–191, 193
Passive classes, 32, 630
Passive components, 311, 437–438
Passive data abstraction object, 301
Passive objects, 32, 622, 630, 666
Patterns, 231, 275–276, 631
Periodic timer components, 435
Periodic use cases, 67
Physical classes, 121
PIM (platform-independent model), 15
Plug-compatible components, 311–313
PLUS (Product Line UML-Based Software Engineering), 14, 43, 309, 666
 integration
 with spiral model, 51–53
 with USDP, 53–57
Ports, 308, 666
 complex, 308
 components, 308–309
 connector, 308
 delegation connector, 311
 provided interface, 308, 599
 required interface, 308, 599

Prerequisite features, 100, 666
 Primary actor, 67–68, 71–72, 667
 Private visibility, 618
 Problem domain objects, 58
 Problem domain static models, 121–122
 Problem-specific static model, 59
 Producer/consumer messages, 258–259
 Producer/consumer problem, 33
 Product family engineering, 43–61, 667
 Product line analysis model, 44
 Product line architecture, 60, 279
 Product line context class diagram, 124, 126, 532, 535, 667
 Product line context model, 124
 Product line engineering, 43–61, 667
 Product line evolution, 161, 324
 Product line kernel software architecture, 50
 Product line scoping, 55, 58, 667
 Product line system, 71, 124, 667
 Product line system class, 126–127
 Product line use case model, 44, 50, 92
 Product line use cases, 72–73
 Product Line UML Based Engineering (PLUS), 667
 Product lines, 3, 667

- analyzing requirements, 60–61
- commonality, 58, 349
- degree of commonality and variability, 60–61
- evolutionary dynamic modeling, 147–148
- kernel system, 148–149
- modeling commonality and variability, 120–121
- relationship between use case model and feature model, 56
- variability, 58

 Product line-specific patterns, 11
 Provided interfaces, 308–311, 434, 599, 667
 Provided port, 308–311, 429, 432–433, 667
 PSM (platform-specific model), 15
 PuLSE (Product Line Software Engineering) method, 10

Q

Queuing model, 667

R

Rational Unified Process (RUP), 667–668. *See also* USDP
 Real-time, 667
 Real-time applications, 251–253
 Real-world objects, 20
 Registration services, 38–39
 Relationships

- aggregation, 112
- aggregation hierarchies, 27–28
- application classes, 136
- associations, 26
- composition relationship, 27–28
- between features and classes, 205
- use cases, 75, 88
- whole/part relationship, 27

 remote method invocation (RMI), 667
 Required interfaces, 308–311, 599, 667
 Required ports, 308–311, 429, 432, 667
 Requirements, 95

- model, 47, 320, 322
- modeling variability, 7–8

 Requirements modeling, 12–13, 47, 58, 95, 628, 667–668
 Requires association, 108
 Requires dependency, 108
 Reusable code components, 4–5
 Reusable components, 44
 Reusable design patterns, 10–11
 Reuse category, 213–217, 668
 Reuse stereotype, 213–217, 668
 Reusing software, 3
 Reverse engineering, 49–50
 Reverse evolutionary engineering, 50, 90–92, 126–127, 668

- context model, 129–130
- entity class models, 131
- product line evolution, 57

 RMI (remote method invocation), 36, 667, 668
 Role category, 213, 668
 Role stereotypes, 213, 668
 RPC (remote procedure call), 36
 RUP (Rational Unified Process), 667, 668. *See also* USDP

- S**
- Scenarios, 146, 668
 - Secondary actors, 67–68, 72, 668
 - Selection condition, 79–81, 516
 - Semaphores, 239, 668
 - Separation of concerns, 290
 - Sequence diagrams, 47, 141, 144–147, 614, 619
 - versus* communication diagrams, 145–146
 - message sequence numbering, 155
 - Sequential application, 32
 - Sequential server, 301, 668
 - Sequential state decomposition, 176
 - Sequential statecharts, 178
 - Server components, 296–297, 300–305, 588
 - Servers, 297, 668
 - Service providers, 38
 - Service registry, 38–39
 - Simple components, 281, 311, 668
 - Simple use cases, 71–72
 - Single systems
 - dynamic analysis, 148
 - dynamic modeling, 142–147
 - use case model, 66–69
 - SOAP (Simple Object Access Protocol), 38
 - Software agents, 266, 452, 454
 - Software application engineering, 44–45, 319–344, 669
 - application analysis modeling, 322
 - application construction, 324
 - application deployment, 325–326, 328, 656
 - application derivation, 319
 - application design modeling, 322
 - application features, 320, 322–324
 - application requirements modeling, 320, 322
 - application testing, 323
 - in electronic commerce product line, 503
 - in factory automation product line, 610–612
 - feature-driven, 323
 - incremental application implementation, 322
 - in microwave oven product line, 329–343
 - phases, 320–323
 - product line artifacts, 328
 - systematic approach, 328–329
 - tradeoffs, 328–329
 - USDP, 57, 323–325
 - Software application system testing, 324
 - Software architectural communication patterns, 231, 640–650, 669
 - Asynchronous Message Communication pattern, 253–254, 640
 - Asynchronous Message Communication with Callback pattern, 255–257, 641
 - Bidirectional Asynchronous Message Communication pattern, 254–255, 642
 - Broadcast pattern, 264, 643
 - broker communication patterns, 260–262
 - Broker Forwarding pattern, 260, 644
 - Broker Handle pattern, 260–262, 645
 - Discovery pattern, 262–263, 646
 - group message communication patterns, 263–266
 - Multicast communication, 264–266, 648
 - negotiated communication patterns, 266–269
 - Negotiation pattern, 266–269, 647
 - product line implications, 269
 - Subscription/Notification pattern, 265–266, 648
 - Synchronous Message Communication with Reply pattern, 255–257, 649
 - Synchronous Message Communication without Reply pattern, 258–259, 650
 - Software architectural patterns, 59, 231, 631, 669
 - applying, 275–277
 - documenting, 275
 - Software architectural structure patterns, 231, 631–639, 669
 - Broker architectural pattern, 245–246, 632
 - Centralized Control architectural pattern, 247–248, 633
 - Client/Agent/Server architectural pattern, 246–247, 634
 - Client/Server architectural pattern, 240–245, 635
 - Communicating Components architectural pattern, 251–253
 - Distributed Control architectural pattern, 248, 636
 - Hierarchical Control architectural pattern, 250, 637
 - implications of control patterns, 250–251
 - Kernel architectural pattern, 239–240, 638
 - Layers of Abstraction pattern, 232–238, 639

- Software Architectural Transaction patterns, 269, 651–653
 - Compound Transaction pattern, 272–273, 651
 - Long-Living Transaction pattern, 273–275, 652
 - Two-Phase Commit Protocol pattern, 270–271, 653
- Software architecture, 34–41, 279–280, 669
 - component-based, 285–289
 - designing, 282–285
 - distributed applications, 282–285
 - in electronic commerce product line, 487–502
 - in factory automation product line, 599–609
 - integrating communication models, 284–285
 - issues, 280
 - message communication between components, 283
 - in microwave oven product line, 418–439
 - structure of components, 282
- Software interface, 307–311
- Software objects, 136
- Software product family, 5–6, 669
- Software product line, 3, 5–6, 669–670
 - analysis modeling, 47–48, 58–59
 - architectural design, 60, 279
 - common components, 4
 - commonality, 6, 58
 - design modeling, 48, 59–60
 - engineering, 43–61, 669–670
 - analysis modeling, 47–48
 - double spiral model, 52–53
 - incremental component implementation, 48
 - phases, 45–49
 - product line testing, 48
 - requirements modeling, 47
 - evolution, 151–155, 670
 - alternative communication diagrams, 152–155
 - kernel communication diagrams, 151
 - optional communication diagrams, 152
 - variant communication diagrams, 155
 - evolutionary dynamic analysis, 147–148
 - feature modeling, 7–8
 - forward evolutionary engineering, 89–90
 - generic architecture, 6
 - requirements modeling, 47, 58
 - reverse evolutionary engineering, 90–92
- Software requirements, 65, 95
- Software reuse, 4–5
- Specialization, 28
- Spiral model, 51–53, 670
- State machine modeling, 12, 169
- State machines, 170, 670
 - Boolean guard condition, 188–189
 - child, 181–182
 - feature condition, 188–189
 - inherited, 169, 181–188
 - parameterized, 169, 188–191
- State transition diagram, 169, 670
- State transition table, 169, 670
- State transitions, 170–171, 620, 670
- Statechart diagrams, 169, 614, 620–621, 670
- Statecharts, 13, 169, 629, 670
 - actions, 174, 620
 - activities, 174–176
 - concurrent, 179
 - conditions, 172, 620
 - events, 171, 620
 - flat, 176
 - hierarchical, 176–180
 - mutually exclusive, 180–181
 - orthogonal, 178–180
 - sequential, 178
 - state transition, 170, 620
 - states, 171, 620–621
 - substates, 176, 621
 - superstate, 176, 621
- State-dependent
 - control classes, 180
 - control objects, 59, 139, 170, 194, 670
 - dynamic analysis, 148, 193–197
 - dynamic modeling, 195, 197
 - interactions, 170, 193
- State-dependent objects, 169–171
- State-dependent variability, 197–199
- States, 171, 620–621, 670
 - feature-dependent, 189
 - hierarchical decomposition, 176–178
 - sequential decomposition, 176

- Static entity class modeling, 130-135
 - Static modeling, 12, 59, 119-120, 670
 - in electronic commerce product line, 451-461, 481
 - in factory automation software product line, 525-535
 - in microwave oven product line, 362-367
 - Static models, 13, 47
 - entity classes, 130-135
 - kernel class, 120
 - optional classes, 50, 120
 - problem domain, 121-122
 - variant classes, 50, 120
 - Stereotypes, 69, 121, 213-217, 626-627, 670
 - application classes, 136
 - definition of, 120
 - external classes, 124, 128-129
 - feature groups, 110
 - features, 107
 - reuse, 213
 - role, 213
 - Structured classes, 307
 - Structuring criteria, 139
 - Subclasses, 28, 206, 207-211
 - Subscription/Notification pattern, 265-266, 304-305, 539-540, 648, 670
 - Substates, 176, 621, 671
 - Subsystem communication diagram, 671
 - Subsystems, 280, 293, 671
 - Superclass, 28, 127
 - Superstate, 176, 621, 671
 - Synchronous (tightly coupled) communication, 36, 255-259
 - Synchronous message communication, 255-259, 671
 - Synchronous Message Communication pattern, 255-259, 595
 - Synchronous message communication with reply, 255, 671
 - Synchronous Message Communication with Reply pattern, 255-257, 283, 424, 486, 596, 649
 - Synchronous message communication without reply, 259, 671
 - Synchronous Message Communication without Reply pattern, 258-259, 424, 429, 650
 - Synchronous messages, 255-257, 623
 - System context class diagram, 124, 671
 - System context diagram, 124
 - System context model, 124, 671
 - System interface object, 138, 671
- T**
- Tables
 - feature groups, 114-115
 - representing features in, 108-110
 - Tagged values, 99, 100-101, 110, 627
 - Target application, 324, 326
 - TCP/IP, 38, 234-236
 - Testing software application system, 324
 - T.H.E. operating system, 233
 - Threads, 32, 622, 671
 - Throwaway prototyping model, 51
 - Tightly coupled message communication, 255-259, 671
 - Tightly coupled message communication with reply, 255, 671
 - Tightly coupled message communication without reply, 258, 671
 - Timer actors, 67-68, 70
 - Timer event, 171, 671
 - Timer object, 139, 672
 - Timing diagram, 672
 - Transactions, 269-275, 672
 - Transition phase, 55, 57
 - Two-phase commit protocol, 270, 672
 - Two-Phase Commit Protocol pattern, 270-271, 653
- U**
- UDDI (Universal Description, Discovery, and Integration) framework, 40
 - UML (Unified Modeling Language), 12, 14, 283, 613-630, 672
 - UML 1.0, 15
 - UML 1.3, 15
 - UML 2.0, 15, 307, 613-630
 - UML notation
 - active objects, 622
 - class diagrams, 616
 - classes, 615
 - concurrent communication diagrams, 622-624

700 Index

- UML notation *continued*
 - deployment diagrams, 625–626
 - interaction diagrams, 618–619
 - objects, 615
 - packages, 622
 - passive objects, 622
 - statechart diagrams, 620–621
 - use case diagrams, 614
 - USDP (Unified Software Development Process),
 - 43, 319, 672
 - Analysis workflow, 54
 - artifacts, 53
 - construction phase, 55, 57
 - Design workflow, 54
 - elaboration phase, 55–57
 - Implementation workflow, 54
 - inception phase, 54
 - integration of PLUS with, 53–57
 - phases, 53
 - Requirements workflow, 54
 - software application engineering, 57, 323–325
 - Test workflow, 54
 - transition phase, 55, 57
 - workflows, 53
 - Use case analysis, 56, 102
 - Use case diagrams, 614, 672
 - Use case modeling, 12, 47, 50, 58, 672
 - actors, 66–69
 - in electronic commerce product line, 443–447
 - in factory automation product line, 510–516
 - in microwave oven product line, 69, 348–356
 - requirements phase, 66
 - single systems, 66–69
 - software product lines, 69–71
 - use cases, 66–67
 - Use case packages, 103–104, 107, 672
 - Use cases, 12–13, 45, 47–48, 66–67, 614, 672
 - abstract, 75
 - alternative or optional sequence of interactions, 78
 - commonality/variability analysis, 89
 - definition of, 58, 66
 - dependencies, 73, 75
 - development strategies, 89–92
 - documenting, 72–73
 - documenting small variations, 75–76
 - extensibility, 75
 - features, 101–107
 - functionality inserted at variation point, 75
 - identifying, 71–72
 - include and extend relationships, 75
 - primary actor, 71
 - questions about, 73
 - relationships, 75, 88
 - reuse, 75
 - scenarios, 146
 - variation points, 73, 75, 102
 - User interface component, 295
 - User interface object, 31, 138, 292, 537, 672
- V**
- Variability, 6, 672
 - abstract classes, 207–211
 - managing, 60
 - product lines, 120–121
 - Variability analysis, 97, 351
 - Variable component, 313–314
 - Variant classes, 59, 120, 122, 215, 217, 672
 - Variant communication diagrams, 155
 - Variant components, 4, 57, 425–426, 672
 - Variant parameterized classes, 216
 - Variant subclasses, 207, 411
 - Variation points, 73, 75, 322, 673
 - abstract classes, 206, 207–211
 - alternative features, 106
 - analyzing impact of, 164–168
 - impact of, 161
 - mandatory alternative, 77
 - optional alternative, 77
 - optional features, 106
 - variant communication diagrams, 155
 - Visibility, 617–618, 673
- W**
- W3C (World Wide Web Consortium), 38
 - Waterfall model, 51

Web services, 38–39, 673
Web services broker, 38, 40
WebLogic, 38
WebSphere, 38
White box testing, 323
White page brokering, 39, 260–262, 673
Whole/part relationships, 27–28, 290, 617, 673
Wrapper components, 40–41, 673
WSDL (Web Services Description Language), 39

X
XML (Extensible Markup Language), 38, 662

Y
Yellow page brokering, 39, 262–263, 673

Z
Zero-or-more-of feature groups, 114, 116, 673
Zero-or-one-of feature groups, 110–112, 673

