

Index

A

Abbyy FineReader, 238
Abstract Factory pattern, 30–31, 348
Abstract instances compartments, 30
Abstract interfaces, 57
Access relationships, 65
Acquaintance, 95, 110–13
Adaptive Communication Environment (ACE) patterns, 110
Addy, E., 189
Adobe Acrobat Capture, 238
Advanced C++ Programming Styles and Idioms (Coplien), 11
Alencar, P., 41
Alexander, Christopher, 11, 87
Alexandrian templates, 52, 114
American College of Radiology (ACR), 266
Analysis, 357
Analysis patterns, 10
Analysis Patterns (Fowler), 341
Anantha, K., 32, 33
Anderson, B., 277
Anderson, Bruce, 11
AOP (aspect-oriented programming), 29
Application interfaces, 55
Application-specific frameworks, 14–15
“Applying Design Patterns in UML” (Garlow, Holmes, and Mowbary), 17
Architectural fragments, 27
Architectural patterns, 78–79

Architecture, 357
Architecture Description Language (ADL), 53
Architecture patterns, 11
ASoC (Advanced Separation of Concerns), 38
Aspect-oriented programming (AOP), 29
Aspect-Oriented Software Development (AOSD)
ASoC (Advanced Separation of Concerns), 38
aspects, 38
SOC (separation of concerns), 38
SOD (Subject-Oriented Design) model, 38
Trace pattern example, 39–40

B

Beck, Kent, 11, 16
Behavior diagrams, 9
Behavioral composition, 1, 4, 17
Behavioral composition techniques
architecture fragments and superimposition, 27–29, 42
contracts, 42–43
LOTOS (Language of Temporal Ordering Specification), 41
OOram method, 21–25, 42. *See also as main heading*
Prolog and model-checking techniques, 41

- Behavioral composition techniques (*cont.*):
 roles and composite design patterns, 25–27
 role/type/class modeling, 29–31. *See also as main heading*
- Bell Canada, 34
- Black box frameworks, 14
- Black box reuse, 48–49
- Blackboards, 11, 51
- Booch, G., 74
- Booch method, 8
- Bosch, Jan, 21, 27, 29, 42, 155, 172
- Bottom-up software development, 357
- Brokers, 11
- Builder pattern, 348–49
- Buschmann, Frank, 8
- C**
- Case studies. *See* DCRM distribution subsystem; DCRM filtering subsystem; DICOM Client Application Entity (AE) subsystem; DICOM UL (Upper-Layer) subsystem; Feedback control framework; Waiting-queue simulation
- “A Case Study in Software Reuse” (Addy, Mili, and Yacoub), 189
- Castellani, X., 43
- Catalysis approach
 defined, 32–33
 explained, 36
 modeling concepts, 37–38
 natural language replacement, 303
 versus POAD, 37
- CBSE (component-based software engineering), 49–50, 240
- Chain of Responsibility pattern, 26, 27
- Clarke, S., 38, 43
- Class associations, 357
- Class diagrams, 9, 51, 97, 141–44
- Class libraries, 357
- Class-level instantiation, 102–03
- Coad and Yourdon method, 8
- Coad, Peter, 12
- Cognitive Science Online Community, 238
- Collaborating classes, 7, 16
- Collaboration diagrams, 9, 74–75
- Collaborations, 37, 74–78, 357
- Command pattern, 349
- Common Object Request Broker Architecture (CORBA), 15, 54
- Component classifications, 99
- Component diagrams, 9–10
- Component granularity, 7
- Component interfaces
 application and platform, 55
 CORBA, 54
 COTS components, 54
 horizontal channels, 55
 Interface Definition Language (IDL), 54
 Module Interconnection Languages (MILs), 53–54
 object-oriented, 55
 vertical channels, 55
- Component-based frameworks, 32, 35–36
- Component-based software engineering (CBSE), 49–50, 240
- Components, 357
- Composite design patterns, 25–27
- “Composite Design Patterns” (Riehle), 26, 42, 342
- Composite pattern, 51, 65–66, 68, 71, 89, 346
- Composition patterns, 38
- “Composition Patterns: An Approach to Designing Reusable Aspects” (Clarke and Walker), 38
- Concrete interfaces, 57
- Concretization, 139–40
- Constraint diagrams, 30
- Constructional design patterns
 black box versus white box reuse, 48–49
 class models, presence of, 51
 defined, 51, 358
 design components, 5, 50–51. *See also as main heading*
 examples, 51
 general schema, 58–59
 interfaces, 51–52, 58–60, 345–49
 modeling, as UML components, 82
 POAD, role played in, 48–49

- structural composition approaches, 32
 - versus structural patterns, 51
 - Contracts, 42–43, 55
 - Coplien, Jim, 11
 - CORBA (Common Object Request Broker Architecture), 15, 54
 - COTS components, 54
 - Counted Pointer, 11
 - Cowan, D., 41
 - “Creating Applications from Components: A Manufacturing Framework Design” (Schmid), 14
 - CSER (Consortium for Software Engineering Research), 34
 - Cunningham, Ward, 11
- D**
- DCOM (Distributed Component Object Model), 15
 - DCRM application
 - components, 240–41
 - hardware setup, 239–40
 - subsystems, 241
 - system architecture, 240
 - DCRM distribution subsystem
 - acquaintance and retrieval, 242–43
 - class diagram, developing initial, 248–50
 - design optimization, 250, 251
 - Detailed Pattern-Level diagram, 245, 246f
 - pattern internals, instantiating, 245–48
 - pattern selection, 243
 - Pattern-Level diagram, 244
 - Pattern-Level with Interfaces diagram, 244–45
 - requirements analysis, 242
 - DCRM filtering subsystem
 - acquaintance and retrieval, 252–53
 - class diagram, developing initial, 260–61
 - design optimization, 261–63
 - Detailed Pattern-Level diagram, 255–56
 - pattern internals, instantiating, 256–60
 - pattern selection, 253
 - Pattern-Level diagram, 253–54
 - Pattern-Level with Interfaces diagram, 254–55
 - requirements analysis, 250, 252
 - Dependency, 358
 - Deployment diagrams, 10
 - “Describing and Composing Patterns Using Role Diagrams” (Riehle), 42
 - Design, 358
 - Design by Contract* (Meyer), 42
 - Design components
 - black box versus white box, 50
 - characteristics, 50–51, 99
 - generic, 32
 - modeling, 5
 - patterns as, 4, 5
 - reusability, 7–8
 - “Design Components: Towards Software Composition at the Design Level” (Keller and Schauer), 32, 43, 97
 - Design composition techniques. *See also* Patterns, composition and integration of and application construction, 3
 - design models, developing supportive, 3, 4
 - patterns, use of, 5
 - Design frameworks. *See* Frameworks
 - Design methodologies, dimensions of, 6
 - Design patterns. *See* Patterns
 - Design Patterns and Contracts* (Jezequel, Train, and Mingins), 342
 - “Design Patterns as Language Constructs” (Bosch), 42
 - Design Patterns: Elements of Object-Oriented Software* (Gamma et al.), 12, 20, 71, 256, 341
 - “Designing Component-Based Frameworks Using Patterns in the UML” (Larsen), 43, 60
 - Detailed Pattern-Level diagrams, 41
 - Detailed Pattern-Level models
 - defined, 63, 68–69
 - design decisions, 70
 - diagrams, creating, 96, 97
 - dynamic aspects, importance of, 130–31
 - example, 70–71, 130, 131f
 - interfaces, realization relationship between, 132

- Detailed Pattern-Level models (*cont.*):
 pattern structure, exploring, 130
 relationships, 69–70
 schematic diagram, 69
 UML syntax support, 70, 130
- Detailed pattern view, 358
- Diaz, R., 54
- DICOM Client Application Entity (AE) subsystem
 acquaintance and retrieval, 288
 class diagram, developing initial, 296–98
 design optimization, 298, 299f
 Detailed Pattern-Level diagram, 292, 293f
 pattern internals, instantiating, 294–96
 pattern selection, 288–90
 Pattern-Level diagram, 290, 291f
 Pattern-Level with Interfaces diagram, 290–92
 requirements analysis, 286–88
- DICOM (Digital Imaging and Communication in Medicine)
 client/server architecture, 269–70
 overview, 266
 scope, 267–69
 specification versus implementation, 266–67
- DICOM UL (Upper-Layer) subsystem
 acquaintance and retrieval, 275–77
 class diagram, developing initial, 286
 communication options, 270
 design optimization, 286
 Detailed Pattern-Level diagram, 283
 pattern internals, instantiating, 284–85
 pattern selection, 277–78, 279–80f
 Pattern-Level diagram, 278, 281–82
 Pattern-Level with Interfaces diagram, 283
 requirements analysis, 271–75, 276f
- Digital Content Remastering (DCRM) application. *See* DCRM application
- Distributed Component Object Model (DCOM), 15
- Distributed information systems, 265
- Distributed medical informatics systems
 functionalities of, 265
 standards, 265–66
- Distributed objects, 8
- Document understanding. *See also* Digital Content Remastering (DCRM) application
 applications, 238
 explained, 237
 systems, 237–39
- “Documenting Frameworks Using Patterns” (Johnson), 25
- Domain, 358
- Domain analysis, 358
- Domain-specific frameworks, 15, 16
- Dong, J., 41
- D’Souza, D., 32, 36, 43
- Dynamic interfaces, 57
- Dynamic models, 10, 22, 24–25
- Dyson, P., 276
- ## E
- EJB (enterprise Java Beans), 15
- Embedded systems, 8
- Embedding, 101
- Encapsulation, 101
- Enterprise Java Beans (EJB), 15
- Executable components, 99
- “Extending Object-Oriented Systems with Roles” (Gottlob, Schrefl, and Rock), 25
- ## F
- Fayad, M., 14
- Feedback control framework
 class diagram, developing initial, 166–67
 code generation, 168
 design optimization, 168–73
 Detailed Pattern-Level diagram, 161–63
 example: quality control in production
 lines, 173–77
 grouping, 168
 implementation, Java code sample, 178–87

- instantiating pattern internals, 163–66
 - overview, 155
 - pattern selection, 158–59
 - Pattern-Level diagram, 159–60
 - Pattern-Level with Interfaces diagram, 160–61
 - reduction, 168
 - requirements analysis, 156–58
 - traceability, 169–70, 172
 - Filters and pipes, 11
 - Finite State Machine (FSM) pattern, 276–77
 - “Finite State Machine Patterns” (Yacoub and Ammar), 277
 - Foote, B., 14
 - Formal specifications, 52–53
 - Fowler, Martin, 10
 - Fragmentation technique, 324–26
 - Framework Adaptive Composition Environment (FACE), 322–24
 - Frameworks
 - application-specific, 14–15
 - black box, 14
 - classifications of, 14–15
 - class-level instantiation, 102–03
 - as collaborating classes, 16
 - component-based, 32, 35–36
 - versus constructional design patterns, 78–79
 - defined, 14
 - documentation of, 8, 16
 - domain-specific, 15, 16
 - GUI, 15
 - hot spots versus fixed spots, 16
 - metapatterns, 12
 - pattern-level instantiation, 102
 - patterns, use in development, 3, 4, 16, 102–03
 - POAD, developing with, 150–51
 - and real-time applications, 16
 - reuse, 15
 - and system design, 15–16
 - white box, 14
 - FSM pattern, 276–77
 - Functional interfaces, 56
- G**
- Gamma, Erich, 11, 12
 - Gang of Five, 12
 - Garlow, J., 17
 - Generalization relationships, 65
 - Generic design components, 32
 - GoF (Gang of Four), 12
 - GoF templates, 52
 - Gottlob, G., 25
 - Granularity, 7, 78–79
 - GUI frameworks, 15
 - Gullekson, G., 95
 - Guruprasad, K., 32, 33
- H**
- Hard-merge operation, 148–49
 - Harel, David, 9
 - Harrison, W., 38
 - Health Level Seven (HL7), 266
 - Helm, R., 12, 42
 - Hierarchical decomposition, 101
 - Hierarchical Object-Oriented Design (HOOD) method, 8, 62
 - Hillside Group, 12
 - Hitz, O., 238
 - Holmes, C., 17
 - HOOD (Hierarchical Object-Oriented Design), 8, 62
 - Hooks/Templates approach, 12
 - Horizontal channels, 55
 - HotDraw architecture, 16
 - Hot-Spot approach, 12
- I**
- Idioms, 11
 - Implementation diagrams, 9
 - Information hiding, 48
 - Ingold, R., 238
 - Inheritance hierarchy, restructuring, 148

Instantiation by realization, 139
 Instantiation by subclassing, 139
 Interaction diagrams, 9
 Interaction-oriented composition, 20
 Interface classes, 59, 66
 Interface Definition Language (IDL), 54
 Interface operations, 59, 66
 Interface specifications, 53
 Interfaces. *See also* Component interfaces
 abstract versus concrete, 57
 and constructional design patterns,
 51–52, 345–49
 defined, 48
 in design components, 50–51
 functional, 56
 multiplicity, 57
 pattern, 58–60
 POAD, use in, 48
 properties, 56–57
 referential, 56
 requester (client) versus provider (server)
 role, 57
 role classification, 56–57
 signature versus behavior, 57
 static versus dynamic, 57

J

Jacobson, Booch, and Rumbaugh method, 8
 Jacobson, I., 74
 Java Foundation Classes (JFC), 15
 Jezequel, J., 43
 Johnson, R., 12, 14, 16, 25

K

Keller, R., 32, 34–35, 43, 97
 Kent, S., 21, 29
 Kleppe, A., 304
 Kristensen, B., 25, 42

L

Lam, S., 54
 Larsen, G., 35–36, 43, 60
 Lauder, A., 21, 29
 Layered object model (LayOM), 28–29, 42
 Liao, S. Y., 43
 Lifecycles
 patterns, 12–14
 software development, 7
 LOTOS (Language of Temporal Ordering
 Specification), 41

M

Martin, Robert, 276
 Matching techniques, 114
 Measurement patterns, 10
 Mechanisms, 74–78
 Mediator pattern, 26, 349
 Metapatterns, 12
 Method, 358
 Methodology, 358
 Meyer, B., 42
 MFC (Microsoft Foundation Classes), 15
 Micro-architectures, 7, 16
 Microsoft Foundation Classes (MFC), 15
 Mili, A., 189
 Mining, for patterns, 8, 12–13
 MixIn inheritance, 148
 Model-View-Controller (MVC) pattern, 20,
 118
 Modularity, 358
 “Module Interconnection Languages” (Diaz
 and Neighbors), 54
 Mowbary, T., 17
 MVC (Model-View-Controller) pattern, 20,
 25

N

National Electrical Manufacturers Associa-
 tion (NEMA), 266

Navigation techniques, 114
 Neighbors, J. M., 54
 Nobel, James, 20
 NRC (National Research Council of Canada), 34
 NSERC (National Sciences and Research Council of Canada), 34

O

Object Constraint Language (OCL), 27, 304–05
The Object Constraint Language: Precise Modeling with UML (Warmer and Kleppe), 305
 Object Windows Library (OWL), 15
 Object-oriented analysis and design. *See* OOAD (object-oriented analysis and design)
An Object-oriented Framework for Feedback Control Applications (Yacoub and Ammar), 102–03
 “An Object-Oriented Framework for Measurement Systems” (Bosch), 42, 155
 “Object-Oriented Reuse: Experience in Developing a Framework for Speech Recognition Applications” (Sirinivasan and Vergo), 17
 Objectory method, 8
Objects, Components, and Frameworks With UML: The Catalysis Approach (D’Souza and Wills), 43, 341
 Observation patterns, 10
 Observer pattern, 26, 27, 28–29, 30, 40, 75, 76, 80, 123–24, 345–46
 Odenthal, G., 35
 OO modeling techniques, 73
 OO programming languages, 4
 OOAD (object-oriented analysis and design)
 interfaces, 55
 methodologies, 8
 models, static and dynamic, 10, 22, 24–25
 Pattern Oriented Technique (POT), 33–34
 patterns, evolution of, 11–12

 and UML, 8–9, 73
 OOPSLA (Object-Oriented Programming, Systems, Languages, and Applications) conferences, 11
 OOram method
 abstraction process, 22
 integration into UML, 42
 modeling process, 21–23
 role models, 21–22
 synthesis process, 21, 23–25
 views provided, 22–23
 OORASS (Object-Oriented Role Analysis and Software Synthesis). *See* OOram method
 Ossher, H., 38
 Osterbye, K., 25, 42
 Overlapping patterns, 88, 90–92
 OWL (Object Windows Library), 15

P

Package diagrams, 35, 80–81
 Packages, 79–81
 Pagel, B., 329, 330
 Parameterized collaboration, 75–76
 Parnas, David, 48
The Pattern Almanac (Rising), 193, 340, 341
 Pattern catalogues and libraries
 Adaptive Communication Environment (ACE), 110
 domain-specific versus general purpose, 110–11, 118
 GoF (Gang of Four) book, 12, 42, 43, 110, 118
 searching, 95
 wide availability of, 3, 47
 Pattern dependency, 358
 Pattern documentation templates, 114
Pattern Hatching: Design Patterns Applied (Vlissides), 342
A Pattern Language (Alexander), 87, 341
 “A Pattern Language of Statechart” (Yacoub and Ammar), 277, 283
 Pattern languages, 353, 358

- Pattern lint tool, 328–29
- Pattern Oriented Technique (POT), 33–34
- Pattern tools
 - CASE tool support, 352
 - code generation tool, IBM's, 327–28
 - fragmentation technique, 324–26
 - Framework Adaptive Composition Environment (FACE), 322–24
 - hooks and templates, 329–30
 - pattern lint tool, 328–29
 - PSiGene (Pattern-Based Simulator Generator), 326–27
 - TogetherSoft Control Center, 330–31
- Pattern-based implementation, 353–54
- Pattern-level instantiation, 102
- Pattern-Level models
 - defined, 63
 - design decisions, 65
 - diagrams, creating, 96, 125
 - example, 65–66
 - large applications, 125
 - multiple instances, 124
 - pattern instance relationships, 124–25
 - pattern instances, type and name, 64, 123–24
 - relationships, 64
 - schematic diagram, 63–64
 - UML syntax support, 65
- Pattern-Level with Interfaces models
 - defined, 63, 66
 - design decisions, 67
 - diagrams, creating, 96
 - example, 68, 128–29
 - interface classes and operations, 66
 - interface relationships, 129
 - pattern instance interfaces, 127–29
 - relationships, 67, 128–29
 - schematic diagram, 66–67
 - UML syntax support, 67–68, 128
- Pattern-Oriented Analysis and Design. *See* POAD (Pattern-Oriented Analysis and Design)
- Pattern-oriented designs, 6, 359
- Pattern-oriented frameworks, 359
- Pattern-Oriented Software Architecture: A Pattern System* (Buschmann et al.), 12, 341
- Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects* (Schmidt et al.), 8, 243
- “A Pattern-Oriented Technique for Software Design” (Ram, Anantha, and Guruprasad), 32, 33
- Patterns. *See also* Constructional design patterns
 - application design, use in, 3, 4, 5, 8
 - architectural, 78–79
 - availability of, 47
 - challenges and problems, 354–55
 - classifications of, 10–11, 19–20
 - defined, 10
 - as design components, 4, 5, 34–35
 - discovery versus invention, 11
 - finding and documenting, 3
 - formal specifications, 52–53, 352–53
 - frameworks, developing, 4, 16, 32, 35–36
 - gluing, 3, 4, 17
 - history, 11–12
 - interface specifications, 53
 - lifecycle, 12–14
 - as mechanisms, 74–78
 - mining, 12–13, 351
 - overlapping, 88, 90–92
 - as packages, 79–81
 - polishing, 13–14
 - recipes, 52
 - refactoring techniques, 33–34
 - reuse, 3, 4, 7–8, 12, 14
 - and software lifecycle, 7–8
 - specifying, 52–53
 - stringing, 88, 89–90, 92
 - structural, 51
 - templates, 351
 - types and use, 47–48
 - visual modeling, 61–62
- Patterns, composition and integration of. *See also* Behavioral composition techniques; Structural composition techniques
 - ad hoc, 337
 - in application development, 3, 4, 5
 - behavioral composition, 4, 19

- classification schema and techniques, 19–20
- gluing, 3, 4, 17
- structural composition, 4, 19, 32–33
- systematic, 337–38
- Patterns for Concurrent and Networked Objects* (Schmidt et al.), 12
- “Patterns Generate Architectures” (Beck and Johnson), 16
- Patterns in Java, A Catalog of Reusable Design Patterns Illustrated with UML* (Grand), 255, 256
- PDF (Portable Document Format), 238
- Phase, 92–93
- Platform interfaces, 55
- PLoP (Pattern Languages of Programs) conferences, 12, 13–14, 115
- PLoPD (Pattern Language of Program Design) books, 12, 14, 115, 341
- POAD analysis phase
 - acquaintance, 110–13
 - acquaintance/retrieval iterations, 118
 - activities, listed, 105
 - database queries, 114
 - large applications, 109
 - matching criteria guidelines, 115–16
 - navigation and matching techniques, 114
 - overall approach, 106
 - pattern selection, 117–19
 - precision and recall, assessing, 115
 - purpose, 106
 - requirements analysis, 107–10
 - requirements analysis/acquaintance iterations, 109, 112
 - retrieval, 113–16
 - and software architecture, 108–09
 - software asset retrieval, 114
 - use cases, 107–08
- POAD characteristics
 - architectural development, 100
 - component-based development, 99–100
 - design reuse, 100–101
 - hierarchical development, 101
 - iterative development, 101
 - library-driven development, 100
 - pattern-driven approach, 99
- POAD design phase
 - activities, listed, 121
 - deliverables, 121–22
 - detailed pattern-level models, constructing, 130–32. *See also* Detailed Pattern-Level models
 - overall approach, 122
 - pattern-level models, constructing, 123–26. *See also* Pattern-Level models
 - pattern-level with interfaces models, constructing, 126–29. *See also* Pattern-Level with Interfaces models
- POAD design refinement phase
 - activities, listed, 135
 - class diagrams, developing initial, 141–44
 - concretization, 139–40
 - deliverables, 135–36
 - design optimization, 145–50
 - hard-merge operation, 148–49
 - inheritance hierarchy, restructuring, 148
 - merging activity, 147–50
 - MixIn inheritance, 148
 - overall approach, 136
 - pattern changes, tracking, 139
 - pattern instances, 142–44
 - pattern interfaces, tracing to pattern internals, 141–42
 - pattern internals, instantiating, 137–41
 - reduction process, 145–47
 - specialization, generic to application-specific, 137–38
 - subclassing and realization, 138–39, 138–39
- POAD models. *See also* UML (Unified Modeling Language)
 - characteristics, 71–72
 - collaborations, 76–78
 - composability, 72
 - defining guidelines, 62–63
 - Detailed Pattern-Level model, 68–71. *See also as main heading*
 - hierarchy, 71
 - HOOD (Hierarchical Object-Oriented Design), 62
 - logical model views, properties of, 82–83

- POAD models (*cont.*):
- packages, UML, 79–81
 - Pattern-Level model, 63–66. *See also as main heading*
 - Pattern-Level with Interfaces model, 66–68. *See also as main heading*
 - traceability, 72, 98, 139
 - UML constructs, compared, 83–84
 - visual modeling, 61–62
- POAD (Pattern-Oriented Analysis and Design)
- applying, 340
 - benefits, 103–04
 - versus CBSE, 50
 - constructional design patterns, role of, 48–49
 - design components, 5, 50–51. *See also as main heading*
 - design issues, 4–5
 - Detailed Pattern-Level diagrams, 41
 - and distributed information systems, 265
 - future trends and improvements, 342–43
 - and interface specification, 53
 - interfaces, 48
 - versus Larsen’s approach, 36
 - limitations, 104
 - methodology characteristics, 338–39
 - as OO design methodology, 5–6, 50
 - purposes, 4
 - reactive systems, designing, 286
 - software reuse, 339–40
 - solutions provided, 5–6
 - tool support, need for, 321–22
 - UML, use of, 10, 73–84
- POAD process
- acquaintance and retrieval, 95, 110–16
 - analysis phase, 95–96. *See also* POAD analysis phase
 - code, generation and synchronization, 97–98
 - design phase, 96. *See also* POAD design phase
 - design refinement phase, 97. *See also* POAD design refinement phase
 - frameworks, developing, 150–51
 - lifecycle legend, 93
 - outlined, 92–97
 - phase, defined, 92–93
 - phases of, 93, 94f
 - step, defined, 93
 - stringing versus overlapping, 87–92
- POAD tool support requirements. *See also*
- Pattern tools
 - historical changes, 334–35
 - modeling levels, 332
 - need for tools, 321–22
 - pattern instantiation, 335
 - pattern repository, 334
 - patterns as design elements, 331
 - traceability, 332–33
 - Polishing patterns, 13–14
 - POSA templates, 52
 - Precision, 115
 - Pree, Wolfgang, 12
 - Print-on-demand (POD) services, 238–39
 - Prolog and model-checking techniques, 41
 - Prototypical pattern applications, 26
 - Provided interfaces, 59
 - Proxy pattern, 11, 347–48
 - PSiGene (Pattern-Based Simulator Generator), 326–27
- Q**
- Quibeldey-Cirkel, K., 35
- R**
- Ram, D., 32, 33
- “Reactor: An Object Behavioral Pattern for Concurrent Event Demultiplexing and Event Handler Dispatching” (Schmidt), 70
- Reactor pattern, 65–66, 68, 70–71, 89, 346–47
- Realization, instantiation by, 139
- Real-Time Object-Oriented Modeling (ROOM), 8, 112, 286
- Real-Time Object-Oriented Modeling* (Selic, Gullekson, and Ward), 95

- Real-time systems, 8, 16
 - Recall, 115
 - Recipes, 52
 - Reduction process, 145–47
 - Reenskaug, Trygve, 21, 24, 25, 42
 - Refactoring techniques, 16, 33–34
 - Referential interfaces, 56
 - Refinement, 37
 - Required interfaces, 59
 - Responsibility, 359
 - Responsibility-driven composition, 20
 - Retrieval, 95, 113–16
 - Reusable artifacts, categorization by granularity, 7
 - Reuse, 3, 4, 7–8, 12, 14, 15, 100–101, 339–40
 - Riehle, Dirk, 21, 23, 25, 26, 42
 - Rising, Linda, 193, 340
 - Rock, B., 25
 - Rohnert, Hans, 8
 - Role, 359
 - Role model, 359
 - “Roles: Conceptual Abstraction Theory and Practice Language Issues” (Kristensen and Osterbye), 25
 - Role/type/class modeling, 31
 - abstract factory example, 31
 - constraint diagrams, 30
 - three-layers modeling technique, 30–31
 - UML class diagrams, amended, 30
 - visual specification approach, 29–30, 31
 - ROOM. *See* Real-Time Object-Oriented Modeling (ROOM)
 - Roussel, N., 238
 - Rule of three, 13
 - Rumbaugh, J., 74
- S**
- Saeki, Motoshi, 41
 - Scenario, 359
 - Schauer, R., 32, 34, 43, 97
 - Schmid, Hans, 14, 16, 155
 - Schmidt, D., 8, 12, 14, 70
 - Schrefl, M., 25
 - SELECT approach, 55
 - Selic, B., 95
 - Sequence diagrams, 9
 - Shankar, A., 54
 - Shlaer and Mellor method, 8
 - Simulation
 - defined, 189
 - waiting-queue. *See* Waiting-queue simulation
 - Singletons (C++), 11
 - Sirinivasan, S., 17
 - SOC (separation of concerns), 38
 - SOD (Subject-Oriented Design), 38
 - Software asset retrieval, 114
 - Software components, 49–50
 - Software development
 - design composition techniques, 3
 - design patterns, role and usefulness of, 3, 7–8
 - frameworks, use of, 15–16
 - lifecycle, 7
 - methodologies, 3–4, 6
 - models, importance of, 61
 - OO cellular communications, 17
 - pattern literature, 341–42
 - pattern mining, benefits of, 8
 - pattern-based, 1, 4, 5
 - refactoring approach, 16
 - speech recognition, 17
 - Software engineering
 - CBSE (component-based software engineering), 49–50
 - design methodologies, dimensions of, 6
 - reusable components, 7–8
 - Software frameworks, 15
 - Software library, 359
 - Soukup, J., 172
 - Specialization, 137–38
 - Specification components, 99
 - “Specifying Frameworks and Design Patterns as Architecture Fragments” (Bosch), 27, 42
 - SPOOL project, 34–35
 - Stal, Michael, 8
 - Standalone applications, 238
 - State patterns, 11

Statecharts, 9
 Static interfaces, 57
 Static models, 10, 22, 24–25
 Step, 93
 Stereotyped packages, 79–81
 Strategy pattern, 11, 147–48, 345
 Stringing patterns, 88, 89–90, 92
 Structural composition, 1, 4, 17, 32–33
 Structural composition techniques

- Aspect-Oriented Software Development (AOSD), 38–41, 43. *See also as main heading*
- Catalysis approach, 36–38, 43. *See also as main heading*
- design macrocomponents, abstraction of, 43
- frameworks, component-based, 35–36
- Pattern Oriented Technique (POT), 33–34
- SPOOL project, 34–35

 Subclassing, instantiation by, 139
 “Subject-Oriented Design” (Clarke et al.), 38
 Subject-Oriented Design (SOD), 38
 “Superimposition: A Component Adaptation Technique” (Bosch), 42
 Superimposition technique, 28–29, 42

T

Tarr, P., 38
 TemplateMethod pattern, 347
 Templates, 52
 “A Theory of Interfaces and Modules—Composition Theorem” (Lam and Shankar), 54
 Three-layers modeling technique, 30–31
The Timeless Way of Building (Alexander), 341
 TogetherSoft Control Center, 330–31
 Top-down software development, 359
 “Towards Pattern-Based Tools” (Pagel and Winter), 329
 Trace pattern, 39–40
 Type patterns, 10
 Types, 37

U

UML metamodels

- abstract syntax, 307–14
- the metamodel layer, 305–06
- metamodel packages, 306–07
- OCL (Object Constraint Language), 304–05
- POAD constructs, adding, 305–07
- semantics, 303–04, 318–19
- well-formedness rules, 314–18

 UML (Unified Modeling Language)

- capabilities, extending, 73
- Catalysis approach, 37
- class diagrams with abstract instances
 - compartment, 30
- collaborations, 74–78
- component diagrams, 36, 82
- design patterns, support for, 73–74
- Detailed Pattern-Level model syntax, 70
- development, 8–9
- fragment and superimposition approach, 29
- frameworks, 79
- generalization relationships, 65
- interface classes, 36
- models supported by, 9–10
- package diagrams, 35
- parameterized collaboration, 75–76
- Pattern-Level model syntax, 65
- Pattern-Level with Interfaces model syntax, 67–68
- and pattern-oriented designs, 5–6
- POAD, use in, 10, 74, 76–78, 83–84
- role diagrams, 27
- stereotyped packages, 79–81
- syntax and semantics capabilities, 9

 “Understanding and Using Patterns in Software Development” (Riehle and Zülighoven), 42
The Unified Modeling Language User Guide (Booch, Rumbaugh, and Jacobson), 74
 Use case diagrams, 9
 Use cases, 107–08

“Using Patterns for Design and Documentation” (Odenthal and Quibeldey-Cirkel), 35

V

Vergo, J., 17
 Vertical channels, 55
 Virtual decomposition, 101
 Visual modeling, 61–62
 Visual modeling/specification approach, 29–30, 31
 Vlissides, J., 12

W

Waiting-queue simulation
 acquaintance and retrieval, 193–94
 application engineering, 191–92
 background and requirements, 189–92
 class diagram, developing initial, 201–06
 code generation, 207
 design optimization, 207–09
 Detailed Pattern-Level diagram, 197
 domain engineering, 190
 implementation, Java code sample, 209–35
 pattern internals, instantiating, 197–201
 pattern selection, 194–95
 Pattern-Level diagram, 195–96

Pattern-Level with Interfaces diagram, 196–97

requirements analysis, 192–93

Walker, R. J., 38
 Ward, P., 95
 Warmer, J., 304
 White box frameworks, 14
 White box reuse, 48
 Wills, A., 32, 36, 43
 Winter, M., 329, 330
 Wirfs-Brock, Wilkerson, and Wiener method, 8
 WISDOM++ application, 238
Working with Objects: The OOram Software Engineering Method (Reenskaug), 341–42

X

XML (eXtensible Markup Language), 238

Y

Yacoub, S., 189

Z

Züllighoven, H., 42

