

Index

A

Abstract classes, relationship of classes to interfaces, 69–72

Actions

- expansion regions, 126–127
- UML version changes, 157

Active classes, 83

Activities, exit, 109

Activity diagrams, 11–12

- actions, expansion regions, 126–127
- basics, 117–119
- decomposing actions, 119–121
- edges, 124–125
- flow final, 127–128
- flows, 124–125
 - Petri Nets, 130
- joins, 118–119
 - specifications, 128–129
- partitions, 120–121, 122
- pins, 125
- requirement analysis, 29
- resources, 130
- signals, 121–123
- times to use, 129–130
- tokens, 124
- transformations, 125–126
- UML version changes, 156–157, 159

Activity state, 109–110

Actors, 99–100, 143–144

Acyclic Dependency Principle, 91

Aggregation, 67–68

Agile development processes, 24–25

- resources, 33

Aliasing, 74

Analysis Patterns, 154

Archetypes, 4

Artifacts, 97–98

- UML version changes, 157

Assertions, 50

- subclassing, 51

Association classes, 78–80

Associations, class properties, 37–38

- bidirectional, 41–43
- immutability *versus* frozen, 154
- qualified, 75–76
- unidirectional, 41

Associative arrays. *See* Qualified associations

Asynchronous messages, 61

Attributes

- class properties, 36–37, 39
- classes, 66–67
- mandatory, 39

Automated regression tests, 22

B

Ball and socket notation, 71

Beck, Kent, CRC cards, 63

Bidirectional associations, 41–43

Blueprints, UML as

- forward engineering, 2–3, 6
- reverse engineering, 3, 6

Booch, Grady, UML history, 7–9

Bound elements, 81–82

Branches, 119

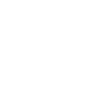
Business use cases, 103

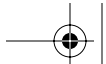
C

CASE (computer-aided software engineering) tools, 3

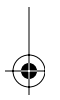
- UML history, 8

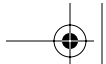
Centralized control of sequence diagramming, 55–57





- Ceremony, agile processes, 25
- Class diagrams, 9, 11–12
 - abstract classes, 69–72
 - active classes, 83
 - aggregation and composition, 67–68
 - association classes, 78–80
 - classifications, 75–76
 - dynamic and multiple, 76–77
 - comments, 46
 - constraint rules, 49–50
 - dependencies, 47–49
 - design, 30
 - documentation, 32
 - generalizations, 45–46, 75–76
 - keywords, 65–66
 - messages, 84–85
 - notes, 46
 - operations, 43–46
 - properties (*See* Class properties)
 - reference objects, 73–74
 - requirement analysis, 29
 - resources, 52
 - responsibilities, 66
 - starting with UML, 16
 - static operations and attributes, 66–67
 - template (parameterized) classes, 81–82
 - times to use, 51–52
 - UML version changes, 158
 - value objects, 74
 - versus* object diagrams, 88
 - visibility, 83–84
- Class properties. *See also* Classes
 - associations, 37–38
 - associations, bidirectional
 - associations, 41–43
 - associations, immutability *versus* frozen, 154
 - associations, qualified, 75–76
 - attributes, 36–37
 - basics, 35–38
 - derived, 68
 - frozen, 72
 - generalizations, 45–46
 - multiplicity, 38–39
 - program interpretations, 39–41
 - read-only, 72
- Class-Responsibility-Collaboration (CRC) cards, 62–63
- Classes. *See* Class properties
 - abstract, 69–72
 - association, 78–80
 - attributes, 66–67
 - Class-Responsibility-Collaboration (CRC) cards, 62–63
 - derivation, 81–82
 - dynamic data types, 153–154
 - generalizations, 35, 36
 - implementation, 153–154
 - presentation, 47
 - static data types, 153–154
 - static *versus* dynamic classifications, 77–78
 - subclassing, 51
 - template (parameterized), 81–82
- Classifications
 - data types, 153–154
 - dynamic and multiple, 76–77
 - implementation classes, 153–154
 - versus* generalizations, 75–76
- Clients/suppliers, 47
- Coad, Peter, UML history, 7
- Cockburn, Alistair, use cases, 105
- Collaboration diagram. *See* Communication diagrams
- Collaborations
 - roles, 143–144
 - sequence diagrams, 144
 - times to use, 146
- Comments in class diagrams, 46
- Common Closure and Reuse Principles, 91
- Common Object Request Broker Architecture (CORBA) standards, 1
- Communication diagrams, 11–12
 - basics, 131–133
 - times to use, 133
- Component diagrams, 11–12
 - basics, 139–141
 - times to use, 141
- Composite structure diagrams, 11–12
 - basics, 135–136
 - times to use, 137
- Composition, 67–68
 - changes between UML versions, 154





- Computer-aided software engineering (CASE) tools, 3
- UML history, 8
- Conceptual perspectives of UML, 5–6
- Concurrent states, 111
- Conditionals, 57–61
 - decisions and merges, 119
- Constraints
 - complete/incomplete, 154
 - rules, 49–50
- Construction, RUP projects, 26
- Continuous integration, 22
- Conventional use, 13–14
- CORBA (Common Object Request Broker Architecture), 1
- CRC (Class-Responsibility-Collaboration) cards, 62–63
- Crystal, agile development process, 24–25
- Cunningham, Ward, CRC cards, 62–63

- D**
- Data tadpoles, 61
- Data types, 74
 - dynamic and multiple classifications, 153–154
 - implementation classes, 153–154
- Decisions, 119
- Dependencies, 47–49
 - keywords, 48–49
 - packages, 91–93
 - resources, 52
 - UML version changes, 155
- Deployment diagrams, 11–12
 - artifacts, 97–98
 - design, 30
 - devices, 97–98
 - execution environments, 97–98
 - nodes, 97–98
 - times to use, 98
- Derivation of classes, 81–82
- Derived properties, class diagrams, 68
- Descriptive rules, UML, 13–14
- Design, 30–31
- Development cases, 25

- Development processes
 - agile, 24–25
 - DSDM (Dynamic Systems Development Method), 24–25
 - Extreme Programming (XP), 22, 24–25, 33
 - fitting processes to projects, 26, 28–29
 - FOD (Feature Driven Development), 24–25
 - iterative, 19–22
 - lightweight, 25
 - Manifesto of Agile Software Development, 24–25
 - Rational Unified Process (RUP), 25
 - resources, 33
 - selecting, 33
 - staged delivery, 21
 - waterfall, 19–22
- Devices, 97–98
- Diagrams
 - activity, 11–12
 - actions, expansion regions, 126–127
 - basics, 117–119
 - decomposing actions, 119–121
 - edges, 124–125
 - flow final, 127–128
 - flows, 124–125
 - flows, Petri Nets, 130
 - joins, 118–119
 - joins, specifications, 128–129
 - partitions, 120–121, 122
 - pins, 125
 - requirement analysis, 29
 - resources, 130
 - signals, 121–123
 - times to use, 129–130
 - tokens, 124
 - transformations, 125–126
 - UML version changes, 156–157, 159
 - basics, 10–12
 - class, 9, 11–12
 - abstract classes, 69–72
 - active classes, 83
 - aggregation and composition, 67–68
 - association classes, 78–80
 - classifications, 75–76





- Diagrams, class, *continued*
 - classifications, dynamic and multiple, 76–77
 - comments, 46
 - constraint rules, 49–50
 - dependencies, 47–49
 - design, 30
 - documentation, 32
 - generalizations, 45–46, 75–76
 - keywords, 48–49, 65–66
 - messages, 84–85
 - notes, 46
 - operations, 43–46
 - properties (*See* Class properties)
 - reference objects, 73–74
 - requirement analysis, 29
 - resources, 52
 - responsibilities, 66
 - starting with UML, 16
 - static operations and attributes, 66–67
 - template (parameterized) classes, 81–82
 - times to use, 51–52
 - UML version changes, 158
 - value objects, 74
 - versus* object diagrams, 88
 - visibility, 83–84
- classifications, 12
- communication, 11–12, 131–133
- component, 11–12, 139–141
- composite structure, 11–12
 - basics, 135–136
 - times to use, 137
- deployment, 11–12
 - artifacts, 97–98
 - design, 30
 - devices, 97–98
 - execution environments, 97–98
 - nodes, 97–98
 - times to use, 98
- interaction
 - basics, 53–56, 147–148
 - CRC cards, 62–63
 - design, 30
 - loops and conditionals, 57–61
 - participants, 53–57
 - sequence diagrams, 53–56
 - synchronous and asynchronous messages, 61
 - times to use, 147, 150
- interactive overview, 11–12
- object, 11–12
 - times to use, 87–88
- package, 11–12
 - basics, 89–91
 - design, 30
 - documentation, 32
 - resources, 95
 - times to use, 95
 - UML version changes, 157
- sequence, 11–12
 - basics, 53–56
 - centralized and distributed control, 55–57
 - collaborations, 144
 - CRC cards, 62–63
 - interaction diagrams, 53–56
 - loops and conditionals, 57–61
 - participants, 53–57
 - returns, 154
 - starting with UML, 16
 - synchronous and asynchronous messages, 61
 - times to use, 61–63
 - UML version changes, 158
- shortcomings, 14–16
- starting point, 16
- state machine, 11–12
 - activity status, 109–110
 - basics, 107–109
 - concurrent states, 111
 - implementing, 111–114
 - initial pseudostate, 107
 - internal activities, 109
 - requirement analysis, 29
 - resources, 115
 - superstates, 110–111
 - times to use, 114–115
 - transitions, 107–108, 111
 - UML version changes, 159
- timing, 11–12
 - basics, 149–150
- types, 11
- types, UML version changes, 157–158

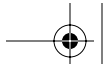




- use case
 - basics, 102–103
 - requirement analysis, 29
 - viewpoints, 6
- Dictionaries. *See* Qualified associations
- Distributed control of sequence
 - diagramming, 55–57
- Do-activities, 110
- Documentation, 31–32
- Domain objects, 47
- DSDM (Dynamic Systems Development Method), 24–25
- Dynamic classifications, 77–78
 - data types, 153–154
- E**
- Edges, 124–125
- Eiffel programming language, 50
- Engineering, forward
 - UML as blueprints, 2–3, 6
 - UML as programming languages, 3
 - UML as sketches, 2
- Entry activities, 109
- Enumerations, 82
- Event switches, 111
- Evolutionary development process. *See* Iterative development process
- Executable UML, 4–5
- Execution environments, 97–98
- Exit activities, 109
- Expansion regions, 126–127
- Extensions, 100–102
- Extreme Programming (XP)
 - agile development process, 24–25
 - resources, 33
 - technical practices, 22
- F**
- Facades, 90–91
- Features of use cases, 104
- Fish-level use cases, 103–104
- Flows, 124–125
 - flow final, 127–128
 - Petri Nets, 130
- FDD (Feature Driven Development), 24–25
- Forks, 117, 119
 - UML version changes, 156

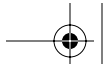
- Forward engineering
 - UML as blueprints, 2–3, 6
 - UML as programming languages, 3
 - UML as sketches, 2
- Found messages, 55
- Frozen property, 72, 154
- Fully qualified names, 89
- G**
- Gang of Four, 27–28
- Generalizations, 35, 36
 - class properties, 45–46
 - sets, 76–77
 - UML version changes, 155
 - versus* classifications, 75–76
- Getting methods, 45
- Graphical modeling languages, 1
- Guarantees, 102
- Guards, 59
- H**
- Hashes. *See* Qualified associations
- History pseudostate, 111–112
- I**
- Implementation classes, data types, 153–154
- Include relationships, 101
- Incremental development process. *See* Iterative development process
- Initial node actions, 117, 119
- Initial pseudostate, 107
- Instance specifications, 87
- Integration, continuous, 22
- Interaction diagrams
 - basics, 53–56, 147–148
 - CRC cards, 62–63
 - design, 30
 - loops and conditionals, 57–61
 - participants, 53–57
 - sequence diagrams, 53–56
 - synchronous and asynchronous messages, 61
 - times to use, 147, 150
- Interaction frames
 - loops and conditionals, 58–59
 - operators, 59
- Interactive overview diagrams, 11–12





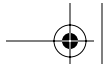
- Interfaces, 65
 - relationship to classes, 69–72
- Internal activities, entry and exit, 109
- Internal activities, exit activities, 109
- Invariants, 51
- Iteration markers, 59
- Iteration retrospective, 28
- Iterations, 20
 - timeboxing, 21–22
- Iterative development process, 19–22
- J**
- Jacobson, Ivar
 - UML history, 7–8
 - use cases, 105
- Jacuzzi development process. *See* Iterative development process
- Joins, 118–119
 - specifications, 128–129
 - UML version changes, 156
- K**
- Keywords, class diagrams, 48–49, 65–66
- Kite-level use cases, 103–104
- L**
- Legacy code, 32
- Lightweight development processes, 25
- Lollipop notation, 71–72, 73
- Loomis, Mary, UML history, 8
- Loops, 57–61
- M**
- Main success scenario, 100–102
- Mandatory attributes, 39
- Manifesto of Agile Software
 - Development, 24–25
- Maps. *See* Qualified associations
- Markers, iteration, 59
- MDA (Model Driven Architecture), 4
- Mellor, Steve
 - Executable UML, 4
 - UML history, 7
- Merges, 119
- Messages, 84–85
 - asynchronous and synchronous, 61
 - class diagrams, 84–85
 - found, 55
 - pseudomessages, 60
- Meta-models
 - definitions, 9–10
 - UML version changes, 157
- Methods
 - implementation of actions, 119
 - versus* operations, 45
- Meyer, Bertrand, Design by Contract, 50
- Model compilers, 4
- Modifiers, 44
- Multiple classifications, 77–78
 - data types, 153–154
- Multiplicity of properties, 38–39
- Multivalued attributes, 39
- N**
- Namespaces, 89
- Navigability arrows, 42
- Nodes, 97–98
- Normative use, 13–14
- Notation
 - ball and socket, 71
 - definitions, 9–10
 - Lollipop, 71–72, 73
- O**
- Object diagrams, 11–12
 - times to use, 87–88
- OCL (Object Constraint Language), 49–50
- Odell, Jim, UML history, 7–8
- OMG (Object Management Group)
 - control of UML, 1
 - MDA (Model Driven Architecture), 4
 - revisions to UML versions, 151–152
 - UML history, 7–9
- OO (object-oriented) programming, 1
 - paradigm shift, 56
- Operations, *versus* methods, 45
- Operators, interaction frames, 59
- Optional attributes, 39
- P**
- Package diagrams, 11–12
 - basics, 89–91
 - design, 30
 - documentation, 32





- resources, 95
- times to use, 95
- UML version changes, 157
- Packages
 - aspects, 93–94
 - Common Closure and Reuse Principles, 91
 - definitions, 89
 - dependencies, 91–93
 - fully qualified names, 89
 - implementing, 94–95
 - namespaces, 89
- Participants, sequence diagrams, 53–57
- Partitions, activity diagrams, 120–121, 122
- Patterns
 - definition, 27–28
 - Separated Interface, 94
 - State, 111–114
 - using, 145
- Petri Nets (flow-oriented techniques), 130
- PIM (Platform Independent Model), 4
- Pins, 125
- Planning, adaptive *versus* predictive, 23–24
- Platform Specific Model (PSM), 4
- Post-conditions, Design by Contract, 50
- Pre-conditions
 - Design by Contract, 50
 - use cases, 102
- Predictive planning, *versus* adaptive planning, 23–24
- Prescriptive rules, UML, 13–14
- Presentation classes, 47
- Private elements, 83
- Profiles, 66
 - UML version changes, 157
- Programming languages, UML as, 3, 5
 - forward engineering, 3
 - MDA (Model Driven Architecture), 4
 - reverse engineering, 3
 - value, 5
- Project retrospective, 28–29
- Properties of classes
 - associations, 37–38
 - bidirectional associations, 41–43
 - qualified, 75–76
 - attributes, 36–37
 - basics, 35–38
 - derived, 68
 - frozen, 72
 - multiplicity, 38–39
 - program interpretations, 39–41
 - read-only, 72
- Protected elements, 83
- Proxy projects, 27
- Pseudomessages, 60
- PSM (Platform Specific Model), 4
- Public elements, 83
- Q**
 - Qualified associations, 75–76
 - Queries, 44
- R**
 - Rational Unified Process (RUP)
 - development cases, 25
 - phases, 25–26
 - resources, 33
 - Read-only property, 72
 - Rebecca Wirfs-Brock, UML history, 7
 - Refactoring, 22
 - Reference objects, 73–74
 - Relationships
 - abstract classes to interfaces, 69–72
 - include, 101–103
 - temporal, 80
 - transitive, 48
 - Releases, 20
 - Requirement Analysis, 29–30
 - Requirements churn, 23
 - Responsibilities of classes, 66
 - Retrospectives
 - iteration, 28
 - project, 28–29
 - Reusable archetypes, 4
 - Reverse engineering
 - UML as blueprints, 3, 6
 - UML as programming languages, 3
 - UML as sketches, 2
 - Revisions by versions (UML)
 - from 0.8 through 2.0, general history, 151–152
 - from 1.0 to 1.1, 153–155





- Revisions by versions (UML), *continued*
from 1.2 to 1.3, 155–157
from 1.3 to 1.4, 157
from 1.4 to 1.5, 157
from 1.x through 2.0, 157–159
- Roles. *See* Actors
- Round-trip tools, 3
- Rumbaugh, Jim
aggregation, 67
composite structures, 137
UML history, 7–9
- RUP (Rational Unified Process)
development cases, 25
phases, 25–26
resources, 33
- S**
- Scenario sets, 99
- Scrum, 24–25
- Sea-level use cases, 103–104
- Searching state, 110
- Separated Interface, 94
- Sequence diagrams, 11–12
basics, 53–56
centralized and distributed control,
55–57
collaborations, 144
CRC cards, 62–63
interaction diagrams, 53–56
loops and conditionals, 57–61
participants, 53–57
returns, 154
starting with UML, 16
synchronous and asynchronous
messages, 61
times to use, 61–63
UML version changes, 158
- Setting methods, 45
- Shlaer, Sally, UML history, 7
- Signals, 121–123
- Single classification, 76–77
implementation classes, 153–154
- Single-valued attributes, 39
- Sketches, UML as, 6
forward engineering, 2
reverse engineering, 2
- Smalltalk, 5
- Software development processes. *See*
Development processes
- Software perspectives, UML, 5–6
- Spiral development process. *See* Iterative
development process
- Stable Abstractions Principle, 92
- Stable Dependencies Principle, 91
- Staged delivery development process, 21
- Standard use, 13–14
- State diagrams. *See* State machine
diagrams
- State machine diagrams, 11–12
activity status, 109–110
basics, 107–109
concurrent states, 111
implementing, 111–114
initial pseudostate, 107
internal activities, 109
requirement analysis, 29
resources, 115
superstates, 110–111
times to use, 114–115
transitions, 107–108, 111
UML version changes, 159
- State tables, 111–112, 114
- Static classifications
implementation classes, 153–154
versus dynamic classifications, 77–78
- Static operations of classes, 66–67
- Stereotypes, 66
- Stories. *See* Features of use cases
- Subactivities, 119–121
- Subclassing, 46
assertions, 51
- Substitutability, 45–46
- Subtypes, 46
- Superstates, 110–111
- Suppliers/clients, 47
- Swim lanes. *See* Partitions
- Synchronous messages, 61
- System use cases, 103
- T**
- Temporal relationships, 80
- Three Amigos, 8
- Time signals, 121
- Timeboxing, 21–22





- Timing diagrams, 11–12
 - basics, 149–150
- Tokens, 124
- Transformations, 125–126
- Transitions, 26, 107–108, 111
 - state, 113
- Transitive relationships, 48
- Trigger, 102
- Types. *See* Data types
- U**
- UML
 - conventional use, 13–14
 - definition, 1
 - descriptive rules, 13–14
 - fitting into processes, 29–32
 - history, 7–9
 - meaning, 14
 - prescriptive rules, 13–14
 - resources, 16–17
 - software and conceptual perspectives, 5–6
 - standards, legal *versus* illegal use, 13–14
- UML as blueprints
 - forward engineering, 2–3, 6
 - reverse engineering, 3, 6
- UML as programming language, 3, 5
 - forward engineering, 3
 - MDA (Model Driven Architecture), 4
 - reverse engineering, 3
 - value, 5
- UML as sketches, 6
 - forward engineering, 2
 - reverse engineering, 2
- UML diagrams. *See* Diagrams and specific diagram types
- UML Distilled*, book editions and corresponding UML versions, 153–155
- UML revisions by versions
 - from 0.8 through 2.0, general history, 151–152
 - from 1.0 to 1.1, 153–155
 - from 1.2 to 1.3, 155–157
 - from 1.3 to 1.4, 157
 - from 1.4 to 1.5, 157
 - from 1.x through 2.0, 157–159
- Unidirectional associations, 41
- Unified Method documentation, 7–8
- Unified Modeling Language. *See* UML
- UP (Unified Process). *See* RUP
- Use case diagrams
 - basics, 102–103
 - requirement analysis, 29
- Use cases
 - actors, 99–100
 - business, 103
 - extensions, 100–102
 - features, 104
 - include relationships, 101–103
 - levels, 103–104
 - MSS (main success scenario), 100–102
 - resources, 105
 - scenario sets, 99
 - times to use, 104–105
 - UML version changes, 155–156
- User Guide*, 115
- User stories. *See* Features of use cases
- V**
- Value objects, 74
- Visibility, 83–84
- W**
- Warehousing systems, Platform
 - Independent Model and Platform Specific Model, 4
- Waterfall development process, 19–22
- Well formed UML
 - definition, 14
 - legal UML, 13–14
- X**
- XP (Extreme Programming)
 - agile development process, 24–25
 - resources, 33
 - technical practices, 22

