Index

A

Abbreviations, for events, 422
ABLE project, 712
Abs association, in business model, 438
Abstract actions, 448, 523
  action sequence diagrams for, 182, 183
  and complex interactions, 154–164
  with exceptions, 335, 337
  and interactions between groups of objects, 154
  and process architecture, 486
  refinement of, 661
  specifying, 336
Abstract action specs, mapped to design action sequences, 639
Abstract business model, for video case study, 570–575
Abstract classes, 150, 662
  types represented by, 149
Abstract connectors, 29
Abstract descriptions, operations on sets in, 573
Abstract diagrams, purposes of, 213
abstract dispatch use case, 440
Abstraction, 34, 37–38, 215, 272, 546, 709. *See also* Refinement
  action, 216, 222–225, 225, 230
  of as-is model, 537
  choosing level of, 556
  and collaboration, 200
  and context models, 593
  defined, 37, 214
  discussed, 10
  documented in business model, 199–200
  and frameworks, 344
  and good documentation, 190–191
  kinds of, 215-230
  model, 216, 220–222, 230
  object, 216, 225–228, 230, 263
  operation, 216, 218–219, 225, 230, 264–265, 268
  orthogonal, 273, 278-279
  precise, 89
  purpose of, 214
  and realization, 157
  and reengineering, 537
  reified and virtual, 229
  theoretical influences on, 709
  varieties of, 217
Abstraction functions, 243, 248
  and behavior specifications, 514
  constructed from component model, 609, 610
  testing with, 241–242
  translation by, 270
  value of, 58
  writing, 555
Abstract models
  bridging gap between implementation and, 611
  clear semantics for, 516
  and collaboration frameworks, 348–349
  frameworks of, 341
  specification for spreadsheet for, 234-235
Abstract objects, 9
  and façade separation, 648
  internal structure of, 256
Abstract specifications, 40
  and recursive refinement, 283
Abstract types, and role decoupling, 672

Access, to variables, 145
Access methods, 482, 641
Account, in video store, 631
Accuracy, 38. *See also* Precision
  and bridge requirements and specifications, 585
  and specification of actions, 509
Action abstractions, 216, 217, 222–225, 230, 254
Action conformance, 254
  checking, 248
  and collaborations, 252–253
  documenting, 249
  and layered design, 250–252
  and object conformance, 254
  and realization of business goals, 247
  within software, 253–254
  testing, 249–250
  vs. operation conformance, 268, 269
Action diagrams, 565, 659
Action exceptions, and composing specs, 331–337
Action implementations, joining, 329–330
Action models, 139
Action occurrence
  and action types, 155
  defined, 80
  pictorial depiction of, 176
  and snapshots, 5
Action parameters, 160
Action postconditions, 13
Action quoting, 142
Action refinement, 544, 661
  and design layers, 252
  exceptions integrated with, 711
  and model refinement, 250–252
  spreadsheet, 247–254
Action Reification, Pattern 14.13, 543, 580
Actions, 3, 7, 254. *See also* Operations
  abstract, 448
  abstracting with, 181–182
  of abstract objects, 257
  accurate specification of, 509
  affecting objects, 4–5
  characterized in documentation, 191
  in collaborations, 172, 253
  concurrent, 168–172, 592
  consistent level of abstraction to, 550
  constituent, 16, 20, 21
  defined, 84
  descriptions by, 167
  designing, 656
  at different scales, 7–8
  directed, 25
  documented, 556
  and effects, 40, 167–168
  explaining effects of, 83
  extension of, 171
  external, 179
  in features, 291
  finding, 549
  finer-grained, 157, 181, 300, 592
  forming implementation of, 661
  in frameworks, 348
  implemented in business model, 439–440
  implementation of, 658
  and interaction models, 153
  internal, 179
  interviewing about, 552
  with invariants, 102–108
  join rules for, 326–327
  joint, 82, 86, 158, 168, 177, 216, 223, 349, 439, 452, 662, 700, 701, 708, 709
  kinds of, 85–86
  localized, 25, 29, 82, 85, 90, 158–159, 177, 258, 260, 662, 708
  and messages, 222–224
  in modeling development process, 521
  and object models, 560
  outputs of, 134–137
  placeholder, 480
  quoted, 122
  raised, 135
  referring to other actions in postconditions, 122
  refinement of, 224–225, 257–258
  reification of, 251–252
  representing objects, 251
  scheduling development of, 540
  sequence diagrams with, 176–177
  simplifying, 119
  snapshots for illustrating, 80, 601, 706
  specifications for, 111-113, 256, 599
  and state, 603
  state transitions as, 127–129
  in templates, 14
  time-triggered, 594

translating state transitions to, 129–130
on UML state transitions, 699
verbs mapped to, 558
vocabulary for describing, 77
zoomed-in, 7, 10
Action sequence diagrams, 176
for abstract actions, 182
design drawn as, 658
Action sequence refinements, state charts in description of, 604
Action specifications, 89, 139, 239, 370
and business rules, 551
clauses, 129
defined, 82
and differences from writing implementation, 109–110
factoring, 117–126
formalizing, 552
implementation of, 329
interpreting, 108–113
joining, 326–328
more precise, 86-92
multiple: two styles, 111–113
parameter types, 111
quoting, 707
and rely and guarantee clauses, 169
separating in documentation, 191
and snapshot pairs, 613
syntax of, 141
and use case, 165
variables used by, 90
Action types, 8
definitions for, 75, 81
in interaction models, 155–156
Active-X, and object locality, 666
ActiveX controls (COM), 402
Actor-level context diagram, 701
Actor roles, separating, 593–594
Actors, 592. *See also* Types
in business model, 201
Actual results, in packages, 316
Ada, and generic types, 148
Adapter classes, and implementation, 245, 246
Adapter objects, 415, 421
Adapters, and object access, 263
addContact action, in requirements spec, 432
addEvent request, 93, 97, 102
addOrderItem action, 439
in requirements spec, 432
ADJ, joint actions and state chart refinement in, 709
ADL. *See* Assertion Definition Language
Administration, and package, 290
"Agent-based" models, 711
Aggregation
and COM composition, 402
and refinement, 231–233
Allen, R., 712
alternateSessions(dates), 118, 119
America, P., 706, 707
"Analysis paralysis," 281
Analysis Patterns (Fowler), 711
Analysts, specifications for, 585
Ancillary tables, 133
Anding
of joining action specifications, 326
of pre/post pairs, 328
type joining, 331
Anecdotes, in narrative of package, 291
Angle brackets ()
for placeholder names, 378
as placeholders for frameworks, 342
Animations and prototypes, for video case study, 620
APIs
early, 394
and legacy components, 651
Appearances
feature, 291
in package, 290
Application architecture, 209, 409, 519, 520, 525, 531
and architectural design, 519, 520
and project evolution, 525
vs. technical architecture, 481, 496–497
Application components, middleware and integration of, 651
Application objects, and reusing user interface, 500
Application program interface, development of, 393
Applications, and locality, 665
Appropriateness, with federated scheme, 408
Architecting, reasons for, 486–490
Architectural design
deliverables of, 519–520
translation-driven approach to, 705
in typical business system, 519–520

Architectural evaluations, and scenarios, 481
Architectural models, 483, 491
  and stakeholders, 487
Architectural packages, 495
Architectural rules and styles, 494–495, 503
  defined, 495, 712
Architectural views, 503
  and physical architecture, 485
  and process architecture, 486
  and software distribution, 485
  and third-party libraries, 493
  various, 483–486
Architecture, 481–503. *See also* Application architecture; Technical architecture
  and abstractions, 38, 181
  aspects to, 515–516
  and code reflection of business model, 509
  component, 409–414
  conceptual integrity of, 490
  consistent patterns used by, 493–495
  CORBA, 403
  database, 520
  decisions and constraints imposed by, 482–483
  defined, 409, 481–482
  definitions by, 412–413
  elements of, 491–493
  federated, 442–443
  heterogenous component, 433
  interframework, 470
  Model-View-Controller, 498
  and objects and databases, 501–502
  pipe-and-filter, 493
  and project planning, 526
  qualities affected by, 487–490
  software, 712, 713
  test, 488
  theoretical influences on, 712–713
  translation-based approach to, 712
  and user interfaces, 498–500
Architecture Definition Language (ADL), 712
Architecture evaluation, with scenarios, 490–491
Architecture implementations
  defined, 412
  and types, 413
Architecture packages, functions of, 416
Architecture types, defining, 416
Arcs, as state changes, 576
Arguments, for output events, 422
Arithmetic types, 376
arrange, 8
Arrows
  in connector diagrams, 418
  for connectors, 404, 411
—A session, 70
As-is elements, analyzing, 546
As-is model
  business, 518, 624
  making, 536-537
  and reengineering, 543
  to-be model informed by, 537
As-is processes, and CRC cards, 549
Aspect-oriented programming, 711
Assemble route, 511–512
Assembly, implementation by, 712
Assertion Definition Language (ADL), and exceptions, 710–711
Assertions, 141
  writing, 267
Assertions and proofs, and package content partitioning, 299
Asset development, 457
assign_mentor action, 87, 89, 92
assignQualified invariant, 71, 74
Association constraints, 73
Association loops, invariants from, 568–569
Association names, and joining, 325
Associations, 5, 57, 88
  across component boundaries, 450
  adornments available for, 61–62
  and business model glossary, 199
  and components, 435
  defined as frameworks, 698
  defined by templates, 370
  defining of, 373
  and interchangeability with attributes, 569
  as inverse attributes, 67
  many–many, 561
  in packages, 286
  as pictorially presented attributes, 220, 221
  pictorial models with, 312
  redundant, 560
  refinement of, 9
  reifying, 561

in state transition matrix, 133
in video store case study, 571
Associative clauses, defined by templates, 370
Assoc-Type framework, 634
Asynchronous calls, 663
Asynchronously invoked actions, 135
Atomic interactions, 523
Attribute definitions, in packages, 286
Attribute ownership, and links, 658, 664
Attributes
for architectural models, 484
in business model, 548
and business model glossary, 199
combining two lists of, 323
const, 73
definitions for, 75
derived, 72, 105
finding, 549
and interchangeability with associations, 569
and invariants, 113–115
inverse, 372
and links, 52, 572
model and reality, 58–59
model frameworks of, 342–346
modeling object's state, 45
in models, 51
and models of state, 698
in packages, 288
parameterized, 45, 59–61, 68, 325, 603, 698, 706
port, 425
retrieving of, from implementation, 220
scheduling development of, 540
and snapshots, 50
and state charts, 127
states as, 126–127
states defined in terms of, 269
in state transition matrix, 133
type model for, 99
visibility of, 531
Attribute types, and legal snapshots, 58
Attribute values
changed to different objects, 53
disallowing combinations of, 67
Authentication, 489
and container, 401
Authorization, 489
and container, 401
Authors attributes, and packages, 288
Auxiliary definitions, in packages for model encapsulation, 310–311
Availability
and container, 401
and runtime qualities, 489
Average operators, 65
Avoid Miracles, Refine the Spec, Pattern 15.12, 531, 533, 582, 611–612
Axioms, 376

B

Bags, in collection types, 689, 691
Basic design, generalizing after, 660
Basic Design pattern (16.8), 531, 533, 654–659, 660, 661, 665, 667
Bass, L., 712
Batch-mode systems, 549
Beads, for connectors, 404, 411
Behavior. *See also* Object behavior
describing, 79
specified with type specification, 323
and type, 33–34
type specification and object, 581
undefined, 92
Behavioral specification, designing component to conform to, 639
Behavior-centric approach, 41
Behavior models, 34, 43, 79–151
and action specification interpretation, 108–113
and actions with invariants, 102–108
factoring action specifications, 117–126
Java implementations of a calendar, 92–97
and more precise action specifications, 86–92
and object behavior, 80–86
Behavior models *(cont.)*
and object types and actions, 706–708
and outputs of actions, 134–137
programming language: classes and types, 143–151
and state charts, 126–133
subjective model: meaning of containment, 137–139

and subtypes and type extension, 113–117
theoretical influences on, 706–708
type specification of, 97–102
and type specifications summary, 139–142
Behavior (object), and state, 46, 47
Behavior specifications, 44
in Catalysis, 514
no overriding, 115
and type models, 140
Beringer, D., 708
Beta version, package-level, 316
Bidirectional properties, 425
Billing package, in BluePhone, 303
Binary components, 39
Binding, 402
Binding times, and components, 389
Black-box components, 455
Black-box composition, 471
Black-box frameworks, 477
Black-box reuse, 642
across component boundaries, 476
Black dots, as starting points, 576
Blank class, 244
BluePhone
decoupling in, 307–308
and multiple imports names, 312–313
and renaming with import, 314–315
BluePhone business overview
horizontal slices, 301–303
vertical slices, 300–301
Böhm's spiral model, 539
Booch, Grady, 705, 710
Booking framework, 635
Boolean attributes, parameterizing, 598
Boolean expressions
in Object Constraint Language, 692
templates presented as, 374
Boolean operators
and invariants, 68
packaging of, 376
Boolean types, 376
Boolean variables, and specifying exceptions, 333
Bottom-up approach, to refinement, 281
Boundary
defining system scope and, 591
and projects, 526
and requirements activity, 518
and system description in collaboration, 592
and system specification, 519
Boundary decisions, in object refinements, 263
Bounded buffer primitive, 492
Brache, A. P., 713
breakLink, 670, 671
Bridge Requirements and Specifications, Pattern 15.2, 581, 585–586
Budget, for reuse, 456. *See also* Costs
Buffers
and architecture, 481
in component for Cat One, 427
Bug reports, in packages, 288, 316
Bugs, and retrieve functions, 150
Build and buildability
and development-time qualities, 488
packages as units of, 285, 315, 317
BuildEnvironment, 317
Build mechanisms, for packages, 526
BuildObject, 317
Build route, 510–511, 512
Business
detailed views separated from abstract views of, 301, 302
and infrastructure components, 652
Business architecture, four-tier, 497–498
Business case, and project evolution, 522
Business collaboration, and business model, 518
Business components, middleware separated from, 650–651
"Business" design, 518
Business domain, type model representing, 558
Business (domain) model, and case study, 528
Business experts, involving, 552–553
Business goals
action conformance and realization of, 247
and program code, 230
representing, 228
Business level of modeling, 35
Business location, and software location, 651
Business logic, application architecture and partitioning of, 496
Business model document, 197
component's context and reengineered, 201
constructing, 201–202

glossary in, 199
Business modeling process patterns, 543–544
Business models, 11–13, 39, 197, 208, 508, 510
building, 36, 505, 543–580
Company Sales, 436
and context model, 594
creating common, 554–555
decoupling in, 306–307
described, 10
documenting, 198–202
drawing, 550
high-level, 300
and horizontal packages, 301
and import of virtual package, 297
initial type model derived from, 597
and items and descriptors, 562–563
making, 275
in packages, 316
and project evolution, 522–523
range of, 550
reengineering requirements for, 537
relationship between system, design and, 625
and requirements activity, 518
retrievals in, 437–439, 443
and rules abstraction, 37
for sales company, 429–432
sources for, 548–549
and static models and invariants, 12
system requirements specifications and, 17
vs. system context model, 624
theoretical influences on building of, 713
Business objects, 543
and persistence, 501
user interfaces connected to, 498
Business organization, pattern for improving, 545–546
Business Process Improvement, Pattern 14.1, 536, 543, 545–547, 548, 661
Business process modeling, 39
Business project, typical project evolution for, 522–526
Business rules, 551
embodied with business objects, 650
implementing, 439–440
within requirements spec, 430–431
Business systems, typical process for, 517–521
Buttons
in small components kit, 405
in user-interface components, 525

C

C++, 143, 649
assert macro with, 267
compile-time template in, 471
contained objects in, 207
dereference use in, 694
and generic types, 148
header files in, 306
implementation mixed with interface with, 474
and include, 293
message implementation by, 415
namespace in, 289
object identity in, 53
and operation specifications, 151
packaging forms in, 318
and public inheritance, 150
pure abstract class in, 147, 219, 672
and runtime type identification (RTTI), 149
template classes in, 148
template List class in, 460
and templates, 477
translations defined in, 426
variables in, 145
Cache framework, 284
Caches and caching, 366
and architecture, 481
patterns, 668
Calendar
and application architecture, 520, 525
Java implementations of, 92–97
type specification of, 97–102
calendarFor operation, 93, 94, 96, 97, 98, 99, 100, 101, 102
Call and return dialog, 25
call action, in video store case study, 577
Callback-styled programming, 458
Call by value, 699
Call patterns
between super-and subportions, 475
between superclasses and subclasses, 477
CallQueue type, 350, 351, 352

Call sequences, 707
cancelCourse action, 118
cancel operation, 125
cancel_reservation action, in video store case study, 577
cancel_reservation use case, in video store case study, 579
cancel_session action, 136
“Capsules”, and theory extensions, 710
*cardinality, meaning of, 63
Cardinality constraints, on connections, 428
Caret marks, for message sending, 422
Case study
  levels of description in, 528–530
  notational tools in, 516
CASE tools, and Catalysis use, 697
Casselman, Rom, 708
Catalysis, 1, 39, 370 *See also* Development process overview; Unified Modeling Language
  abstraction techniques in, 272
  actions characterized in, 222
  application overview of, 505
  attribute, role, and state in, 699
  attributes and operations in, 698–699
  behavior characterized in, 219
  benefits overview for, 40–41
  build function in, 317
  business modeling with, 11–13
  classes joined in, 330–331
  collaboration design with, 347
  collaboration diagram in, 175
  component architectures viewed by, 414
  component specification in, 520
  components within, 18–23
  composing descriptions in, 321
  composing types in, 322
  consistency rules with, 513
  const annotation in, 698
  design approach of, 452
  development layers within, 10–11
  and development process, 31–32
  enumeration types in, 702
  exceptions specifying in, 702
  extending Object Constraint Language in, 694–695
  features overview of, 40
  frameworks repository of, 557
  implementation coverage in, 532
  import and dependencies in, 296–297
  importing packages in, 292–293
  layering of decisions in, 235
  model frameworks as templates, 13–14
  model frameworks for extensibility in, 698
  modeling constructs plus frameworks of, 32–35
  modeling levels of, 35–36
  modeling process in, 521–522
  and multiple routes through method, 510–512
  Object Constraint Language (OCL) in, 693
  object-oriented design, 30–31
  objects and actions in, 3–6
  package scope in, 693
  packages in, 286, 289, 515, 699, 700
  parameter assumptions in frameworks of, 148
  parameterized type notation in, 700
  precise, formal notation in, 290
  precise abstraction with, 214
  principles of, 37–39
  process overview for, 39
  project planning in, 526
  quality assurance in, 514
  and refinement: objects and actions at different scales, 6–10
  refinement use in, 166, 167, 213, 484, 639
  relationships between models and documents defined in, 512
  and requirements specification models, 16–18
  requirements specs in, 255
  responsibilities assigned with, 24–29
  rules for packages and imports in, 298
  sets and flat sets in, 699
  specifying exceptions in, 333
  standard packages in, 378
  states in state chart in, 269
  and subjective models of containment, 698
  support, tools, services, and experiences in, 703
  testing in, 488
  theoretical influences on applying, 713
  type names in, 308, 313
  type system of, 700
  web site for, 703
  zooming in on software: system context, 15–16
  zooming into, 228–229
catalysis.cpp.lang package, 378
catalysis.eiffel.lang package, 378

catalysis.java.lang package, 378
catalysis.smalltalk.lang package, 378
Cat One
  defining, 414–415
  example of component architecture, 415–416
  sample component kit in, 492
Cat One components
  connecting, 426–428
  specifying, 421–425
Cause, split from effect, 355
Cell abstract class, 244
Cells
  setting as sum of two others, 247, 248
  in spreadsheet, 676
change_dates action, 103, 128
Change_dates operation, defining, 105
changedDetails action, 626
Change Detect, in components for Cat One, 427
Change requests, in packages, 288, 316
Changes
  impact of defined in Catalysis, 515
  management of, 214
Channels, and communication components, 525
Cheng, J. H., 706
Choose a Level of Abstraction, Pattern 14.6, 556–557
Circular extensions, 296, 304
C language
  dereference use in, 694
  include in, 286
Class box, 48
Class class, Java instantiation of, 398
Class diagrams, 48
  and code, 207–208
  and component implementations, 206, 207
  importance of, 260–261
Class(es), 195, 196, 662. *See also* Types
  in code package, 318
  collaboration review for, 658
  as component?, 390–391
  concrete, 672
  defined, 143
  extension of, 146
  generic, 341
  hierarchies of, 477
  implementing with, 143
  instantiation of, 147, 148
  Instructor, 71
  joining, 330–331
  localized actions added to, 260
  in packages, 287
  partial abstract, 304
  program dependency on, 265
  pure abstract, 304
  role-based decoupling of, 304–306
  specifications, 149–151
  specification types, design types and, 654–656
  templates as generic, 364
  vs. type, 114–115
  types implemented by, 31, 149
Classes specifications
  and constructors, 150
  and implements assertions, 149–150
  and operations specifications, 151
  and retrieval, 150–151
Class frameworks, “vertical” interfaces of, 476
Class inheritance, 402
  and package import, 293
Class layer of design, 143
Class libraries, UI frameworks in, 460
Class templates, 341
Class variables, 144
Clients
  model interpretations for, 613–615
  and requirements documents, 193–194
  terms natural to, 71
  type intersection and multiple, 323
  understanding terminology of, 548
Client-server architecture, 516
Client-server-styled component systems, 393
Clients tier, in four-tier business architecture, 497
CLU, 672
Clustering, and container, 401
Cobol
  variable name in, 308
Cockburn, A., 713
CoCreateInstance (COM), 402
Code
  abstracting single operation in, 162–164
  abstract models represented in, 241
  and analysis, 281
  and class diagram, 206, 207–208, 260

and components specifications, 583
defensive checks in, 332–333
documentation and lack of, 187–188
frameworks: specs to, 465–470
and inheritance, 472
invariants in, 71–72
legacy, 535
models and designs: business to, 508–510
and modifications in package, 288
and operation refinement, 219
and operation spec conformance, 266
operation specs compared with, 267–268
and optimization, 667
in packages, 287, 289
and platform independence, 649
real world to, 228
specification as, 244
specification directly translated to, 244
specify, document, implement, and test: business to, 510
and specs, 230
testing by representing specification in, 243–246
Code changes, and encapsulation, 312
Code frameworks
alternative ways to compose, 469–470
combining, 468–470
Code reuse, 453
framework approach to, 461–465, 477
Code separations, of interfaces from implementations, 319
Code-sharing, 146
Coherence principles, 658
Coleman, D., 705, 706, 713
Collaboration abstractions, 200, 215
Collaboration-based modeling, and OORAM approach, 705
Collaboration diagrams, 124, 261
in business model, 201
and interactions between objects, 661
in Unified Modeling Language, 701
Collaboration frameworks, 346–352
applying, 350–352
invariant effects used in, 348–349
reusable, 18
Collaboration models, and context models, 592
Collaboration refinements, 136, 215
Collaborations, 40, 172, 252, 478, 516. *See also* Actions; Objects
abstracting with, 181–182
action conformance and, 252–253
for architectural models, 484
in business (domain) model, 528
and business model glossary, 199
careful design of, 347
and context models, 591
defined, 23
encapsulated, 173, 174, 178
in features, 291
as first-class entities, 709
high-level business, 300
and implementation and integration of classes or components, 510
and interaction among groups of objects, 32–33
and interaction models, 153
interactions shown by, 653
and internal design of component, 36
among objects, 216
open, 173, 177–178, 179
in packages, 288
plugging together, 358, 359
principals coupled to, 480
recurring patterns of, 509
refinement of, 247
for refining requirement, 354–355
responsibilities stated for, 662
and reuse, 276
and review for classes, 658
as scoping construct for invariants, 708
situations for use of, 154
specifications for, 179–182
summary of, 182
and system action specifications, 600
systems built from, 358–359
and technical architecture model, 525
uses of, 173–178
Collaborations and Responsibilities, Pattern 16.10, 531, 661–663
Collaboration specifications
abstracting with collaborations and actions, 181–182
defined, 181
external actions, 179
internal actions, 179

invariants, 179–180
sequence constraints, 180–181
Collection functions
in Object Constraint Language, 689, 690
Collection Galore's collections, 482
Collections, 63–65
and precise specifications, 90–91
Colllection operators, 69
Colored shapes
copying, 369
Colors and highlights, 524
COM, 520, 649
components connected with, 414
composition mechanisms with, 402
and design rules for technical architecture, 524
directed actions implemented in, 25
interfaces with, 401, 402
interoperating with CORBA, 413
and Microsoft Web site, 712
COM+ (Microsoft), 387, 498
components with, 401–403
merits of, 403
and persistence, 392
standards, 383
Combinational logic gates, and architecture, 491
Command-line parser, 496
Commits, of major transactions in video store system, 616
Common Object Request Broken Architecture. *See* CORBA
Communication
across boundaries of organizations, 554
components, 525
middleware, 496
Company Sales business model, 436
Compiled code, and reusable artifacts, 455
Component architecture, 411
areas for, 29
in Catalysis, 415
and Cat One, 414–428
implementation of, 412
input properties in, 423–424
output properties in, 423
taxonomy of types in, 412–414
Component architecture model, 412
Component architecture type (or style), defined, 412
Component-based design, 26, 148. *See also* Object-oriented design
defined, 410
detailed, 688
for video case study, 679–688
vs. object-oriented design, 391–392
Component-based development (CBD), 41, 80, 81, 280, 385, 397, 712
areas within, 31
components and binding times, 389
defined, 385
and framework approach to building systems, 465
and general components, 385–386
and implementation components, 386–389
importance of specifying interfaces in, 509
Java's technical base for, 401
overview of, 384–392
Component-based modeling, 386
Component-based solutions, for heterogenous components, 432–440
Component building tools, 408–409
Component class, 392
Component connectors
kinds of, 412
and technical architecture, 525
Component/connector technologies, 477
Component design, 11, 39, 197
Component design document, 197
Component diagrams, 418
unfolded version of, 421
Component frameworks, 27–28
Component implementations, 18, 505
for business system, 517
and classes and roles in design of, 206–208
designing to meet a specification, 639–640
documenting, 206–208
inside of, 520–521
internal and external views of, 206
patterns for, 641–678
theoretical influences on, 713
trade-offs with, 639
in typical business system, 520–521
Component instance, 392
defined, 390
Component1 instance, on component diagram, 418
Component interactions, old vs. new styles of, 393–394

Component kits, 28–29
and architecture, 11, 491–492
and component building tools, 408–409
graphical user interface kit, 404–405
of large components, 406–408
pluggable components library, 404–409
of small components, 405–406
Component layer, *vs.* object layer, 30
Component library, 446, 447
Component models, in case study, 529
Component packages, 445
contents of, 387
interface definitions in, 391
Component partitioning, 397
Component-port-connector model, 388, 410–412, 705
component connector, 411–412
example connectors, 412
Components, 18–23, 202, 255, 383, 451. *See also* Implementations
and architecture, 409–414, 491–492, 493
built with façades, 324
built with Java, 398–401
in Catalysis, 414
Cat One, 421–425
in code, defined, 387
and collaboration, 23
with COM+, 401–403
connecting, 397, 428
connection protocols between, 299
and connector-based pluggability, 477
contents of, 386
with CORBA, 403–404
cross-component links, 435, 437
dynamically creating and connecting, 428
evolution of, 392–398
extracting generic code, 444–445
and federated architecture, 442–443
general, defined, 386
generalizing, 455, 634
glue, 441–442, 443
hardware, 590
heterogenous, 383, 428–443, 452, 511, 512, 610
horizontal layering of, 643
implementation of, 36, 386–389
as interacting objects, 508
interaction protocols between, 14
interface of, 178
internal, 195
legacy or third-party, 450–451
lower interfaces for customizing of, 459–460
models of constituent, 433–435
in models to designs: business to code, 508
and objects, 392
partitioning model between, 21–23
and persistence, 392
and plug conformance, 449
pluggability, 323, 388, 395, 444
and postcondition retrieval, 23
roles played by, 21, 201
software built from, 322
specifications for, 36, 123, 131, 421, 505
and standardization, 395–397
theoretical influences on, 712
third-party, 437, 441, 450
two versions of system design, 19–20
vs. objects, 390
why move to?, 397–398
Component specifications, 18, 36, 197, 208
for business systems, 517
and design process, 587
documenting, 197, 202–206
glossary in, 205
modeling and designing in, 507
in packages, 316
patterns for, 530–531, 581–615
and recursive decomposition, 589
state charts in, 204
theoretical influences on, 713
writing, 205–206
Component technology, 712
and infrastructure components, 652
Component type, 392
ComponentType.new, 428
Component views
composing, 609–610
specifying, 607–608
Componentware management, 446–447
Component writers, and architecture, 413
Compose Component Views, Pattern 15.11, 582, 587, 609–610
Composing models and specifications, 321–337
action exceptions and composing specs, 331–337

combining packages and their definitions, 324–331
joining and subtyping, 322–324
sticking pieces together, 321–322
Composite, 590
Composition
binding times for, 389
and design styles for frameworks, 465
and refinement, 229
Concept map(s)
and business model, 518
for video store case study, 569, 570
Conceptual integrity
of architecture, 490
Conceptual problems
identifying, 583
Concrete classes, 672
and inheritance, 474
Concrete import, 297
Concrete initialization
in packages, 316
Concrete packages, 297, 298
and virtual imports, 297–298
Concrete review
and interpreting models for clients, 613
Concurrency
and abstractions, 38
in design, 708
and object-oriented databases, 502
and threads, 492–493
and use cases, 164
Concurrent actions, 168–172
defined, 592
exploring interference, 170–171
and granularity, 169–170
meaning of postconditions, 171–172
and rely and guarantee, 169, 708
Condition variables, 493
Configuration, and user interfaces, 498, 499
Configuration management
and packages, 285, 318
and separation of business rules, 551
confirmed Session, 126
confirmMember action, 626
Confirm()method, 55
confirmOrder action, 439
in requirements spec, 432
Conformance
action, 247
defined, 215
documenting of, 230, 264, 611
and documenting operation spec, 266–268
implementation, 247
justification, 362
plug, 449
and refinement, 213
rules of, 710
Congruent equality, 367
connect action, between ports, 416
Connections, design of, 643
Connector design, for Cat One, 417–418
Connector diagrams, 418
interpreting for Cat One, 418
Connector notations, components and, 452
Connectors, 477
abstract, 29
and architecture, 416, 491–492, 493
bidirectional, 405
in Catalysis, 414
and components, 36, 40, 525
between component instantiations, 404
Connectors *(cont.)*
constrained, 424–425
defined, 410
implementing in Cat One, 415
and kits, 410
between large components, 406, 407
notation, 411
theoretical influences on, 712
transaction, 425
transfer, 425
Connector specification, for Cat One, 416–417
Connector standards, 395, 397
Connector technology, 712
Consolidation stage, in development spiral, 513
Const attribute, 73
Constituent, 590
Constituent components, models of, 433–435
Constrained connectors, 424–425
Constraints
flushing out useful, 568
on state, 77

uniqueness, 695
Construct a System Behavior Spec, Pattern 15.7, 531, 533, 582, 587, 596–599
Constructors, 150
ConsultingEngagement type, 114
Contact managers, and application architecture, 520, 525
Contacts system, type model for, 434
Container, for JavaBeans, 401
Container packages, 700
Containment
and COM composition, 402
subjective models of, 137–139, 698, 708
and variables, 145
Content abstract class, 244
Context, and control of invariant, 107–108
Context analysis, 587
Context Model with Use Cases, Pattern 15.5, 582
Context models, 592
and scoped connection to business model, 594
Context operators, and invariants, 70
Context symbols, 287
Continuity (or seamlessness), 274
of notation, 198
from problem domain to code, 44
Contradictions, in packages, 296
Contravariance, 266, 267
Controller pattern (CP), subsystem, 495
Control objects, 251
for component-level action, 656
Convenience attributes
added to static model, 551
defined, 119
to simplify specifications, 118–119
and state charts, 633
Cook, S., 271, 705, 706, 707, 709, 710, 713
Cooperation, and multithreaded systems, 492
Copying
of objects, 369
templates for, 366–369
Copy state chart, for video store case study, 633
CORBA (Common Object Request Broker Architecture), 384, 410, 411, 649
and adapters, 263
and architecture, 482
and component architecture, 29
components connected with, 403–404, 414
directed actions implemented in, 25
integration of Java and, 401
interoperating with COM, 413
and object locality, 666
and OMG Web site, 712
parts of, 403–404
and persistence, 392
standards, 383
CORBA bus, 403
CORBA Facilities, horizontal and vertical, 404
CORBA IIOP, 498
CORBA Services, 404
Costs
of developing high quality infrastructure components, 650
of large system development, 652
and object technology, 446
and reuse, 454
in software development, 505
Counter, inc and dec events tracked by, 427
Coupling, 641. *See also* Decoupling
of input properties to output properties, 423
minimizing, 662–663
and optimization, 667
reducing, 674
CourseEngagement type, 114
Covariant join semantics, and component views, 610
CRC cards, and as-is processes, 549
CRC (classes, responsibilities, collaborations) technique, 346, 662
Creating a Common Business Model, Pattern 14.5, 554–555
credit action, for video store, 631
Credit cards processing application, orthogonal abstractions and refinement with, 278
Customer-business relationship, in video case study, 570–571
Customer definition, in video store case study, 572
Customer identifier (CID), 437
Customization
and architecture, 490
and lower interfaces, 459–460
of OCL with Catalysis extension, 695
of process patterns, 530
Cut-and-paste, *vs.* import, 454
Cycle time, 546

D

Data access services, 496
Database architecture, and architectural design, 520
Database components, descriptions of, 525
Databases
  hybrid object-relational, 502
  object-oriented, 502
  relational, 501–502, 650
Data-centric approach, 41
Data Checker, 20, 21
DataRange type, 102
Data storage and access, and infrastructure components, 652
Date type, immutable or mutable models of, 60–61
DCOM (Microsoft), 410, 411
  and object locality, 666
Debugging
  and defensive programming, 333
  of implementations, 709
  of invariants, 72
  and operation specifications, 151
  and retrieval, 150
  and retrieve functions, 150
Decision points, 129
Decisions, documenting for project, 522
Declarations, in packages, 290, 318
Declarative access, 501
Decline of the American Programmer, The (Yourdon), 187
Decompositions, 283
  recursive, 509
Decoupling, 237, 264, 643, 663
  and basic design generalizing, 660
  of design, 3, 23, 27, 154, 146, 347, 589
  and designing to spec, 640
  of documentation, 189
  and flexibility, 25–29, 277
  and operation refinement, 265–266
  and packages, 285, 303–307
  role, 672–673, 676
  and single façade object, 262
  strong, 265–266
Decoupling pattern (16.1), 641–642
Deep design, 655, 656
DeepThought approach, and operation spec conformance, 266, 267
Default, defining, 372
Default arguments, and explicit template parameters, 366
Default implementation, delegated to interfaces, 474
Default notation, template defining meaning of, 372, 373
Defensive programming, *vs.* design by contract, 332–333
"Defensive specification," 333
Defined behaviors, 332
Definitions. *See also* Dictionary; Glossary
  combining from different sources, 322
  in documents, 191
  and features, 291
  in frameworks, 347
  generic, 148–149
  importance of, 75
  importation of, 294, 337, 362
  joining, 321, 324–325
  in model frameworks, 379
  and model interpretation for clients, 614
Definitions. *(cont.)*
  in packages, 286, 292, 295
  precision for, 708
  renaming of, 314
  of subtypes, 328
Delegation
  and design styles for frameworks, 465
  role, 478
  via polymorphic interface, 474
delete action, 626
Delete() event, 97
delete(Reservation), 681
Deliverables, 587
  and component views, 607
  and system specifications, 626
delivered Session, 126
Dependencies
  Catalysis import and, 296–297
  layering of, 664
  and packages, 18, 285, 295, 296, 303, 319
  reducing, 265–266, 564, 641
Dependency diagrams, 642
  drawing, 304
Deployment descriptor, in Enterprise JavaBeans component, 401
Deployments/deployment phase

and business changes, 533
and project evolution, 526
Derivation, 145
Derived attributes, 72, 105
Derived invariants, 105
Derived parameterized attributes, 72
Derived specifications, 105, 707
Design. *See also* Object-oriented design
and abstraction tools, 181
and architecture, 409, 481, 502
background to guidelines for, 713
basic, 654–659
and business (domain) model, 528
Catalysis approach to, 452
classes and roles in, 206–207
collaboration descriptions of, 173
of collaborations, 182, 346–347
component, 11, 39, 197
in component package, 387
component *vs.* object layer within, 30
concurrency in, 708
decoupled, 3, 23, 27, 146, 154, 347, 663
and decoupling pattern, 641
deep, 655, 656
detailed, 688
encapsulated, 178
end-product, 590
enterprise-level, 40
focus of, 517–518
generic component, 447
and good documentation, 230
and heterogenous components, 432
high-integrity, 40
of high-level component design, 643
inheritance-based, 472
interface-centric, 397
internal, 36, 208
line between implementation/test and, 532
and meeting specifications, 639–640
and model frameworks, 339
notation, 185–186
object, 11
of object collaborations, 153–154
for object models, 257
in packages, 289
and platform independence, 649
and plug-points, 679
polymorphic, 473
and precision, 69, 74
promoting practical, 589
recursive use-case-driven, 644–645
refactoring of, 521
reification in, 251
and relationships between elements, 465
and reuse culture, 456
seamless, 276
separation of responsibilities in, 333
shortcuts in, 283
and state charts, 604
and systems built from collaborations, 358–359
traditional *versus* framework-style, 465
type, collaboration, refinement, plus frameworks, 32–35
Design action sequences, mapped to abstract action specs, 639
Design by contract, *vs.* defensive programming, 332–333
Design decisions, separating important from less important, 583, 584
Design documents, *vs.* specs, 207
Design elements, tagging, 495
Designers
and design patterns, 669
and notation, 516
and packages, 304
Design layers, and action refinement, 252
Design models, relationship between business, system and, 625
Design patterns, 698
detailed, 669
in packages, 286
Design projects, specifications and implementation in, 196–197
Design reviews, 34, 515
and refinement, 611
Design rules for technical architecture, and project evolution, 524–525
Design team, 195
Design-time qualities, 490
Design types, 174, 662
classes, specification types, and, 654
states of spec types distinguished from, 270–272
vs. specification types, 123–124, 707

Detailed objects, 9
Detailed Tasks package, in BluePhone, 303
Developers, specifications for, 585, 586
Development. *See also* Design; Documents and documentation
  asset, 457
  component-based approach to, 397
  layers within, 40
  and packages, 526
  product, 457
Development cycles, 457
  and refinement, 280
Development layers, 10–11
  abstraction and redesign applied to, 537
  maintained in step, 275
Development process overview, 31–32
  general notes on process, 510–521
  main process patterns, 530–542
  model, design, implement, and test-recursively, 507–510
  objects and actions in modeling of, 521
  and reuse, 453–457
  typical package structure, 526–530
  typical project evolution, 522–526
Development product, package in, 285
Development standards, for project, 525–526
Development team, and documentation, 192
Development-time qualities, and architecture, 481, 487–488
D gate, in component for Cat One, 427
Diagrams, 370, 532. *See also* Snapshots
  action, 565, 659
  action sequence, 576, 658
  collaboration, 661, 701
  component, 418
  and composing models, 322
  connector, 418
  and definitions, 74
  dependency, 642
  interaction, 521, 593, 613, 618, 657, 686, 688
  narrative, 613
  object, 418
  object instance, 628
  package, 652
  in packages, 288, 290
  recursive type, 566
  role-activity, 713
  state transition, 498
  system context, 518, 621–626
  templates presented as, 374
  type, 565, 568
  type boxes in, 324
Dialects, 371, 377
  defined, 372
  examples of semantic rules for, 372–373
  and stereotypes, 371–372
Diamond symbols, 156, 231, 232
  and abstract actions, 224
  "assembly," 249
  in refinements/refinement relationships, 226, 575
Dictionary, 45, 376, 377, 558, 614
  in class diagrams, 260
  defined and described, 74–75
  and formally defined types, 76
Dictionary *(cont.)*
  and formal model, 706
  in packages, 290, 291, 318
  responsibilities detailed in, 661, 662
  states added to, 605
  for video store case study, 571, 572, 577
  writing, 101
Dijsktra, , 706, 709
Directed actions, 25
Directed arrows, in interaction diagrams, 593
Directory, and horizontal standards, 396
Disco
  composition in, 710
  and interaction models, 708
  joint actions and state chart refinement in, 707, 709
Dispatch interfaces, 402
display method, and OOP frameworks, 461
Distributed component frameworks, 403
Distributed transaction management, and container, 401
Distribution
  decisions regarding, 658
  and platform independence, 649
DLL, 384, 387
Documents and documentation. *See also* Writing
  abstraction in, 190–191
  and bridge requirements and specifications, 585
  and business model glossary, 558–559
  and business models, 198–202, 208

clear, 280
and client's terminology, 548
and component implementations, 206–208
and component specifications, 202–206
and continuity of notation, 198
of critical information, 299
decoupled, 189, 208
definitions separated and joined in, 191
degree of formality in, 243
and design, 187–191
effective, 49, 708–709
extra time in writing, 584
factoring models for, 191–192
and framework refinement, 355–357
glossary in, 190
good approaches to, 189–190
internal design in, 208
kinds of, 195–197
lack of, 187
and maintenance team, 192–193
and models and design, 510
with packages, 288, 290, 515
and pictorial design, 188
of project standards, 526
purpose of, 185–186, 193, 208
and reaching audience, 192–195
requirements, 586
snapshots in, 77
and structure, 189, 208
system or component specification, 208
theoretical influences on, 708–709
timing of, 194–195
and type specification, 581
wall-sized, 188–189
wrong approaches to, 187–189
Domain-independent components, 497
Domain-independent facilities, and technical architecture, 496
Domain models, 285. *See also* Business models
Domain objects
and components, 397
implementations of, 124
Domains
and object-oriented design, 43
and package content partitioning, 299
Domain-specific modeling patterns, 698
Domain view, 484
Dot (".") operator, on collection, 65
Double-dispatching, 674
DSDM, discussions about, 713
D'Souza, D., 705, 706, 707, 708, 709, 710, 711, 712
Duplicator component, 425
Dynamic behavior, and equality-like relations, 368
Dynamic binding, 276
Dynamic invariants, 702
and snapshot pairs, 613
Dynamic linking, 393
and plug-ins, 459
Dynamic models, 300
Dynamic name resolutions, 667
Dynamic part, of models, 43

E

Editable packages, 316
Editor_implementation type, 174
Editor's operations, 131
effect construct, reason for, 355
Effect invariants, 707
and batch-mode systems, 549
and behavior models, 107
and business rules, 551
external, 180
on external section of collaboration, 348
and test specifications, 515
and use case, 165
Effect postconditions, writing, 120–121
Effects, 79
and actions, 167–168, 701
common postconditions factored by, 119–120
defined, 120
defined by action, 122
differences between operations and, 709
invariant, 107, 348, 448, 549, 707
joint, 167
localized, 167
named, 120, 129
scheduling development of, 540
specified abstractly, 125–126
split from cause, 355
and success indicator, 334

and test specifications, 515
writing, 120–121
Effort attributes, and packages, 288
Eiffel language, 135, 143
and containment, 145
and generic types, 148
and joining classes, 330
operation specifications in, 151
and pre- and postconditions use, 706
Elements
and architectural views, 503
architecture built on defined, 491–493
naming, 308
storage, 491
in user interface, 614
Elliptical bubbles, in action sequence diagram, 176
Embedded software, 40
Empire State Building. *See* Phased development
Encapsulated auxiliary definitions, model decoupling with, 310
Encapsulated collaboration, 173, 179
and type implementation, 174
Encapsulated designs, 178
Encapsulation, 237, 246, 385
and basic design, 655–656, 657, 659
of class dependency, 266
and implementations, 122
mutual, 391
and optimization, 667
and packages, 285, 294, 310–312
of variability, 444
Enclosure rule, 138
End-product design, 590
End users, business model involvement by, 552
enquire(copy), 681
Enterprise JavaBeans (EJB), 398, 400–401
container use by, 403
CORBA aligned with, 404
in four-tier business architecture, 498
and JAR file, 401
Enterprise JavaBeans pattern, 494
Enterprise-level design, 40
Enterprise-level knowledge management, and business rules, 551
Enterprise-scale server components, 401
Entity-relational (E-R) models, 549, 560
Enumerated types, 65
Enumeration object type, 101
Envy
class synthesis from partial definitions in, 331
prerequisite in, 293
for Smalltalk, 710
Equality
defined, 367
object, 53
templates for, 366–369
type-defined, 367–368
“Equality-like” relation, 367, 368
Equality relationships, defining, 53
equals, definition of, 424
equivShapesEq operator, 369
Error rate, 546
Essential models. *See* Business models
EventContainer interface, 95
Event logging, 496
“Event” messages, 415
Event notification design pattern, 493–494
Event object type, 101
Event-property-method connections, 701
Event protocols, across frameworks, 477
Events, 412, 413
abbreviations for, 422
and architecture, 481
connecting, 426
and delete(), 97
deletion of, 101, 102
and JavaBeans, 399
and ports, 415
of two components, 404
UML definition of, 702
eventsBetween, 97
Exception case, *vs.* undefined case, 332
Exception handling
and composing specs, 331
specifying/documenting of, 333
Exception indication mechanisms, 335
Exception names, 333
Exception paths, defined via refinement, 700
Exceptions, 211, 321
and Assertion Definition Language, 710–711
and composing specs, 328
with general actions, 335–336
raising, 334

required, 332
specifying, 333–337
and use case templates, 336–337
Exception signaling, 496
Exception/transaction abort scheme, 425
Executable code, in component package, 387
Executable communicating state-machine models, 705
Expected results, in packages, 316
Explicit call and return, connectors supporting, 397
Explicit external actions, 179
Explicit postconditions, 111
Explicit preconditions, 111
Explicit substitutions, and stereotypes, 371
Explicit template parameters, 366
Explicit unique key, 53
Exporters, 292
Extends relationship, and use case, 166
Extensibility, with frameworks, 35, 41
Extensible object structures, modeling, 566–567
Extensions, 145, 146, 297
of action, 171
circular, 296
to classes in features, 291
defined by templates, 370
and importing, 292
uses for, 294
External actions
for collaborations, 179, 708
and collaboration specifications, 179
"External" design, 518
External design reviews, 195
External effect invariants, 180, 349
External Reason identifier, 682
External review, *vs.* internal review, 614–615
External specifications, in component packages, 387
External views
in component implementations, 206
separating from internal views, 613
Exception outcomes, writing, 334

F

Façades, 229, 676
components built using, 324
between components in same platform, 648
designing, 656
and high-level component design, 643
multiple, 262–263
and object access, 261–263
roles of, 27
separating, 646–648
symbol for, 21
as transparent media, 647
and use cases, 644
Factories, Pattern 16.16, 660, 674–675
Factoring
of action specifications, 117–126
importance of careful, 709
and operation specs, 203–204
to specification types, 124
specifying effects abstractly, 125–126
Factoring of code, framework *vs.* traditional approach to reuse, 461–463, 464
Factory
and decoupling, 265
to localize creation of new objects, 474
Factory class, 675
Factory-floor schedulers, and application architecture, 520, 525
Factory objects, 31
Facts and rules, in package, 290
Failover behaviors, and Enterprise JavaBean tier, 498
Failover mechanism, 496
Failure conditions, 333
Failure indications, guaranteed, 334, 335
False & b operator, 68
Family trees, and recursive composite pattern, 567
Fan-out, and component architectures, 414
Fault Management, in BluePhone, 302
Feature appearance, in package, 291
Features, in dictionary, 291
Federated architecture, 442–443
Federated schemes, benefits of, 407–408
Federation, benefits of, 443
Filled arrow, for property connector, 418
"Film strip" metaphor, 707
Final class, 146
findCustomer action, in requirements spec, 432
findOrderdispatches action, 440
findProduct action, in requirements spec, 432
Fine-grained actions, 157, 181, 300, 592

Fine-grained use cases, 167
Flag variables, 678
Flat files, 501
  and objects, 501
Flat sets, 91
  and Catalysis, 699
  and object model navigation, 706
"Flat" state diagrams, 126
Flexibility, 154
  constraints, 276
  and decoupling, 25–29, 277
  with federated scheme, 407
  improvements in, 278
  and partitioning components, 639
  and two-way links, 670
Flowcharts, and recursive composite pattern, 567
FOOP
  contributions from, 711
  and framework provisions, 711
  generic parameters of, 148
Footnotes, in narrative of package, 291
Foreign keys, 437
Forest project (Sun), 710
Formal definitions, 291
Formal descriptions, 68–70
  purpose of, 70
Formal diagrams, in packages, 291
Formal functional requirements models, 17
Formalism, and life expectancy of product, 514
Formalization
  and state charts, 605
  and vocabulary, 602
Formalized model, 569–570
  and dictionary, 706
  of video case study, 570
Formalized refined use case specs, for video business, 577–579
Formal justification, 272
Formally defined types, and dictionary, 76
Formal notation, 189
Formal postconditions, and system action specification, 600
Formal preconditions, and system action specification, 600
Forté, 408, 649
Forwarding
  explicit, 477
  and polymorphism, 473
Forward slash (/)
  for derived attributes, 72
  for postconditions, 576
Fountain, and spiral model, 513
Fowler, Martin, 705, 711, 713
Frames, in artificial intelligence subculture, 274
Framework applications, 342
  defined, 344
  unfolding, 351
Framework models, generic, 444
Framework provisions, 711
Frameworks, 32, 35, 703. *See also* Collaboration frameworks
  in abstract problem descriptions, 279
  benefits of, 14
  and code reuse, 461–465
  collaboration, 673
  combining independently designed, 470
  component architecture defined with, 701
  components generalized into, 634
  and componentware management, 447
  composing, 357–358
  defined, 340
  and defining basic types in OCL, 695
  distributed component, 403
  documentation of, 206
  four-tier business architecture as, 498
  generic, 524
  generic, reusable models and designs, 34–35
  granularity of, 352
  implementing for resource allocation, 471
  intuitive behavior with, 700
  models built from, 322, 448
  and new modeling constructs, 211
  and non-object-oriented programming, 465
  nonplaceholder types in, 346
  and packages, 286, 288
  parameter assumptions in, 149
  and placeholders, 179
  for products and items, 635
  products of, 380
  and project evolution, 524
  and recursive refinement, 283
  refining, 352–357
  for specification *versus* implementation, 466

specs to code, 465–470
and stereotypes, 698
unfolding, 343–344
using one to build another, 352
Framework specifications, mirrored by collaborating components, 480
Framework style reuse, *vs.* traditional style reuse, 461–465
FTP transfer, components connected with, 414
Functionality
and runtime qualities, 488
and system architecture, 503
Functional requirements, and requirements activity, 518
Function calls, 29
Functions, 134
Fusion, 705, 706
Future primitive, 493
Futures, and connector standards, 397

G

"Galloping generalization," 457
Gamma, E., 265, 266, 473, 674, 678, 711, 712, 713
Garbage collection
with COM+, 402
Garlan, D., 712
Genders, of ports, 415, 417
General actions, exceptions with, 335–336
Generalization, 654
and basic design, 659
and Catalysis import, 296–297
of design, 641
and insights into model, 564
and objects, 672
Generalize after Basic Design, Pattern 16.9, 660
Generalize and Specialize, Pattern 14.10, 564–565
Generalized reservation, framework for, 635, 636
Generic classes, 341
Generic code components, extracting, 444–445
Generic collaboration, 34
framework, 411
Generic components, and plug-points, 457–460
Generic design elements, in architecture, 409
Generic frameworks, 524
Get_and_set_methods, 641
get methods, 482, 483
GetX() method, and architecture, 482
Global attributes, 66
Glossary
in business (domain) models, 528
in business model documents, 199
in component specifications, 205
creating, 71
in documentation, 190
project, 48
and project evolution, 523
type model as, 558–559
Glue, writing, 388
Glue components, 441–442, 443
Goals, of use case, 165
GoF book (Gamma), 711
Goguen, J. A., 711
Goldberg, A., 713
Golden Rule, of object-oriented design, 611
Golden Rule versus Other Optimizations, The, Pattern 6.2, 273, 276–277, 667
Gosling, J., 710
Graceful degradation, with federated scheme, 407
Graham, on requirements elicitation, 202
Granularity, 165, 169–170, 571
and architectural decisions, 482–483
and components, 392, 393, 397
and context models, 593
of framework, 352
with large components, 408
and refinement, 515
and use cases, 701
Graph form, of interaction diagram, 175
Graphical User Interfaces (GUIs), 205, 625, 650, 676
adapter in, 246
component kit for, 404–405
as connector, 414
construction of, 263
and façade separation, 646
and recursive composite patterns, 566
and system specifications, 595
wizard for, 404
"Gray-box" specifications, 707
Gregor, K., 711
Gries, D., 707
Group, ungrouping of, 368

Guarantee and rely, 169
guarantee clauses, 168
  and process architecture, 486
Guarantee conditions, 264, 265
  anding, 326
Guards, in state charts, 633
GUI. *See* Graphical User Interface
Guttag, J., 710, 711

H

Hardware components, 590
Hardware installation, 526
Hardware platforms, and technical architecture, 520
Harrison, W., 708
Head objects, 174, 177, 657
  and subsystems, 493
Heavy analysis phase, 32
Helm, Richard, 707, 711
Heterogenous components, 383, 428–443, 452, 511, 512, 610
  and component-based solution, 432–440
  and federated architecture, 443–443
  and glue components, 441–442
  and requirements specs, 429–432
  summary of, 443
High-integrity design, 40
High-Level Component Design, Pattern 16.2, 643
High-level design diagrams, 229
High-level system design, within component-based design, 683–687
hire action, in video store case study, 577, 629, 630, 631
hireFromCold, in video store case study, 579
hireFromReservation effect, in video store case study, 579
hire use case, in video store case study, 579
Hoare, , 219, 706, 709
Hold information
  reservations with, 636–637
Holt, A. W., 713
Home interface, in Enterprise JavaBeans component, 401
Horizontal bars, in action sequence diagrams, 176
Horizontal cycles, 540
Horizontal (infrastructure) standards, 396
Horizontal layering, 646
Horizontal services, of technical architecture, 513
Horizontal slices. *See also* Vertical slices
  and package content partitioning, 299, 301, 319
  and packages, 301–303
  of programs, 188
  of technical architecture model, 525
Horizontal standards, 395
Hotel class, 472, 473
HotelGuest protocol, 147, 148
Housekeeping message, and two-way links, 658
HP, and UML, 697
Hybrid object-relational databases, and architecture, 502
Hypertext links, in storyboards, 619

I

ICON Computing, and UML, 697
Identity
  of object, 53
  in object model, 560
IDL
  COM interfaces defined in, 402
  CORBA, 403
ID tags, 55
If...then (==>), 89
Illegal snapshots, 66–68
Illustrations, in narrative of package, 291
Immutable types, 60
Implementation components
  composition of, 387–399
  inheritance, 390
  provided and required interfaces, 388
  separation of interfaces from implementation, 388
  unit of independent delivery in, 388
Implementations. *See also* Classes; Component implementations; Specifications
  adapters for, 245
  by assembly, 712
  attributes retrieved from, 220
  bridging gap between abstract model and, 611
  business world, 56
  code for spreadsheet, 236
  code separations of interface from, 319

of components, 36, 39, 505
conformant, 247
debugging, 709
defined, 215
distributed, 684
and encapsulation, 122, 311
interfaces separated from, 515
Java for calendar, 92–97
line between design and, 532
of links, 658
of object state, 54–56
overriding of, 115
plug-in, 27–28
private operations of, 106
prohibiting multiple inheritance of, 145
recursive across business or domain model, 508
and recursive decomposition, 589
relational database, 55
with retrievals, 244
reuse of, 455
snapshots in, 248
and specifications, 196, 510
for spreadsheet, 236–237
and system requirements specifications, 17
of technical architecture, 497
testing, 268
type use in, 124, 174
Implementations documents, defined and discussed, 195
implements assertions, 149–150
Implement Technical Architecture, Pattern 16.7, 531, 533, 652–653
Implicit event propagation, connectors with, 397
Implicit unique keys, 53
Import, extension, defined, 294
Import, usage, defined, 295
Import diagrams, 292
Imported definitions, extending, 285
Imported packages, renaming, 314–315
Imports and importing
benefits of using, 298
and Catalysis import and dependencies, 296–297
concrete, 297
vs. cut-and-paste, 454
designations for, 292
to extend or specialize, 292–294
of implementations, 329
and joining narrative, 331
name conflicts and multiple, 312–315
name mapping on, 314–315
and packages, 18, 34, 292–298, 319
for private usage, 294–295
rules on, 295–296
and substitution, 365
of types, 305
unfolding, 307
use of, 298–303
virtual and concrete packages and, 297–298
Import statements, 292
Import usage, with packages, 285, 286
Incremental cycles, 513, 514
Increments, and project plan, 526
Independent delivery, of components, 388
Indeterminate state charts, 132
Induction rule, and progressions, 374
Induction template, 375
Inference rules, 711
template packages to represent, 374–375
templates for, 375
Informal definitions, 291
Informal descriptions, 514
Informal justification, 272
Informal sketches, 208
Informal specifications, reviewing, 613
Informal use case templates, 164
Infrastructure components
design, implementation, and rules for, 652
Infrastructure services, 460
Inheritance, 145, 146, 214
and coupling, 28
good uses for, 474
public, 150
replaced with type-based composition, 475–476
and subtypes, 113
and template method, 472–473
Inheritance design
and enhanced interaction diagram, 462
Inheritance frameworks, white-box nature of, 476
Initial type model, business model adopted as, 597–598
Input events, 426
specifying, 421
Input parameters, 134

types required for, 124
Input ports, 415
inputPort.source, 425
Input property
driven by output property, 425
specification of, 423–424
Inputs
constraints on, 424
and decoupling, 265
and outputs, 161–162
parameters as, 160–161
and system actions, 600
and system specifications, 587
Inside
and architectural design, 519–520
and component implementation, 520–521
and projects, 526
and requirements activity, 518
separated from outside and boundary, 526, 528
Inspections and metrics, 526
Instance creation, and coupling reduction, 674
Instance diagrams, 50
Instance snapshots, loops on, 566
Instance variables, 144
in Java implementation, 54–55
Instantaneous event models, 270n9
Instantiating
components, 428
new objects, 265
Instantiation
of class, 147, 148, 674
and factory method, 642
and kits, 410
Instructor class, 71
Instructors, assigning to courses, 71
Integers, in OCL, 692
Interaction diagrams, 259, 260, 613
and action occurrence, 176
and basic design, 657
and collaboration uses, 175–176
for component implementation, 521
and joint actions, 701
and scenarios, 177, 593
support for, 618
for video business system, 686, 688
Interaction models, 43, 153–183
actions abstract complex interactions, 154–164
and actions and effects, 167–168
Interaction models *(cont.)*
and collaborations, 172, 182
and collaboration specifications, 179–182
and concurrent actions, 168–172
designing object collaborations, 153–154
theoretical influences on, 708
use cases are joint actions, 164–167
uses of collaborations, 173–178
Interactions
for architectural models, 484
overall schemes of, 38
protocol of, 154–155
and state, 46
Interception, with COM+, 403
Interchange models, 29
Interface packages pattern, 494
Interface repository, and horizontal standards, 396
Interfaces
black-box typed, 475–476
class implementations of, 147
and component views, 607, 608
and decoupling, 641
default implementation with inheritance plug-points for, 474
dispatch, 402
to external systems, 541
in Java, 672
language-independent access of, 398
lower, 459–460
minimizing, 642
pluggable, 44–445, 480
separated from implementations, 515, 542
between superclass and its subclasses, 708
types represented by, 149
upper, 458–459
Interface standards, COM, 402
Interference
exploring, 170–171
and multithreaded systems, 492
Interframework architecture, 470
Internal actions
and collaborations, 179, 708
in collaborations, 179
and collaboration specifications, 179

Internal components, 195
Internal design, 209
  application architecture and detailed design and technical patterns, 531
  in documentation, 208
  implementation of, 532
Internal design interactions, for Java implementations of calendar, 93, 95
Internal details, for external object behavior abstractions, 97
Internal objects, direct external access to, 261, 262
Internal review, *vs.* external review, 614–615
Internal views, in component implementations, 206
Interoperability, and development process, 453
Interpreting Models for Clients, Pattern 15.13, 547, 582, 613–615
Interviewing, 549, 552
  client's staff, 202
  domain experts, 11
  prospective users, 205
Introspection, 399
int type, 568
Invariant effects, 254, 448
  defined, 107
  used in collaboration frameworks, 348–349
Invariants, 553. *See also* Effect invariants; Static invariants
  and action specifications, 117–118, 324
  actions with, 102–108
  and attributes, 113–115
  and Boolean operators, 68
  in business models, 202
  and business rules, 551
  in code, 71–72
  code vs. business, 70–71
  and collaborations, 253
  and collaboration specifications, 179–180
  common uses of, 72–73
  context and control of, 107–108
  and context operator, 70
  definitions for, 75
  derived, 105
  dynamic, 613, 702
  effect, 107, 348, 549, 707
  and joining static models, 325
  maintenance of, 661
  and model of state, 698
  operation specification elements specified by, 104, 105
  range of, 106
  and require condition, 424
  and requirements management, 586
  and rigor, 514
  scheduling development of, 540
  separating, 191
  simplifying, 119
  and state charts, 126, 127
  states as, 126–127
  static, 45, 66–73, 106, 107, 117, 514, 568, 594, 702
  and static models, 12–13
  and subtypes, 114
  supervision of, 262
  writing, 67
Invariants from Association Loops, Pattern 14.12, 568–569
inv effect
  within business rules in requirements specs, 430, 431
  Trade Supply collaboration, 356
Inverse attributes, 372
inv Instructor, 72
Invocable operations, *vs.* state abstractions (attributes), 119
Invocations, and joining action implementations, 329–330
invoiced Session, 126
Invoking
  of joint actions, 168
  of localized actions, 159
inv Session, 70, 72
isException query, 334
isFree event, 97, 99
isFree operation, 93, 98
Item descriptors, framework for, 635
Items and Descriptors, Pattern 14.9, 562–563
Iterations, 530
  and development cycles, 513, 514
  and package structures, 540
  and project plans, 526
IUnknown interface (COM), 402

J

Jackson, D., 707, 710, 713
Jackson, M., 706, 707, 709
Jacobsen, I., 705, 708
JAD. *See* Joint-Application Development
JAR files, packaging with, 400
Java, 135, 143, 649
  calendar implementations in, 92–97
  class objects in, 149
  components built with, 398–401
  and containment, 145
  extended interface in, 147
  import in, 286
  inner class in, 699
  interfaces in, 147, 219, 304, 672
  interface support with, 474
  object identity in, 53
  objects and attributes represented in, 56
  and operation specifications, 151
  packages and packaging in, 289, 318, 710
  and persistence, 392
  scaled packages in, 710
  serialization with, 501
  support for types with, 147
  types represented in, 149
  virtual machine with, 649
  visual building tools with, 408
JavaBeans, 383, 387, 398, 410, 414, 520
  basic, 399–400
  CORBA aligned with, 404
  and design rules for technical architecture, 524
  event model, 480
  improved components with, 400
  and persistence, 392
  Sun's, 411
  vetoing with, 470
  Website for, 712
Java class, and Java interface, 532
Java components, in JAR files, 400
Java implementation, of object state, 54–55
Java RMI, 498
Javascript (on Netscape), 649
Java's Enterprise JavaBeans (EJB) standard, 400, 401
Java Transaction Service, and CORBA model, 404
Java virtual machine, 460
Java Workshop (Sun), 408
Job class, 471
Johnson, R., 711
Join composition, 448
Joining, 322
  of action implementations, 329–330
  of action specifications, 326–328
  classes, 330–331
  definitions, 191, 321, 325
  narrative, 331
  packages, 325
  static models, 325–326
  type definitions, 328
  type specifications, 325, 328–329
Joining and subtyping, type intersection: combining views, 323–324
join mechanism, and UML, 697
join relationship, purpose of, 207
join rules, 700
Joins, and definitions, 324–325
Joint action occurrences, in interaction diagrams, 593
Joint actions, 82, 86, 158, 168, 177, 216, 223, 439, 452, 662, 700, 708, 709
  in collaboration frameworks, 349
  to localized actions, 160–161
  between stateful objects, 707
  use cases as, 164–167
  vs. localized actions, 158–159
Joint-Application Development (JAD) sessions
  and project evolution, 523
Joint effects, 167
Jones, Cliff B., 267, 706, 708
Jordan, M., 710
Justification, 379
  and architectural models, 484
  and conformance, 362
  and refinement claims, 510
  and refinements, 272
  and system architecture: high-level design, 529

K

Kerberos, 489
Keys, 53
  for types in entity model, 560

Key-value pairs, 416
Kit architecture, 31, 40
Kits
  component, 404
  and components crossing projects, 444
  problems with components not designed as, 443
  of small components, 405–406
Knuth, D., 708
Knuth's "Web" of literate programming, 185, 710
Kurki-Suonio, R., 707, 708
Kurth, , 710

L

Labels
  association, 57
  for attributes, 50
  in snapshots, 48
Lambda calculus, 699
Language, specialized, for communication models and designs, 516
Language-specific call-return convention, 701
Language-specific exception mechanisms, 702
Lano, Kevin, 709
Larch language, 288
  and framework provisions, 711
  and package use, 710
  renaming and importing of traits, 711
Large business components, 406–408
Large components, kit of, 406–408
Large/larger-grained components, 383, 392, 393
  design of, 643
Large-grained objects, reified use cases into, 645
Large-grained reuse, 456
Larman, Craig, 713
Layered architectures, notation for, 495
Layering, via packages and templates, 377, 378
Lea, Doug, 705, 708, 712
Legacy adapter, in four-tier business architecture, 498
Legacy code, and reengineering pattern, 535
Legacy (or third-party) components
  driving, 651
  using, 450–451
Legacy systems
  business components integrated with, 650
  and refinement, 701
  reviewing contents of, 591
Length instance variables, 220, 221
Libraries, of reusable abstract specification frameworks, 467
Library business models, 550
LibraryManagementSystem: return action, 601, 602
Link and Attribute Ownership, Pattern 16.11, 531, 534, 664, 665, 667
Link implementations, and object locality, 665–666
Links
  and attribute ownership, 658
  and attributes, 52, 572
  in collaboration frameworks, 349
  cross-component, 435, 437
  directionality of, 531, 664
  implementing, 658
  in loops, 569
  many-many, 671
  one-to-one, 671
  redundant, 560
  in snapshots, 48
  two-way, 670–671, 673
Liskov, B., 710
Listboxes, 453
list members, 670
Lists, 376, 377
  in user-interface components, 525
Literate Programming (Knuth), 708
Load balancing, and EJB tier, 498
Locality, object, 665
Localized actions, 25, 29, 82, 85, 90, 158–159, 177, 708
  joint actions to, 160–161
  vs. joint actions, 158–159
Localized effects, 167
Localized specifications, 139
Local objects, plus cross-component links via events, 469
Local variables, 144
Lodging services collaboration, 177–178, 180
Logic
  temporal, 707

three-valued, 706
"Lollipop" notation, 701
Loop invariants, finding, 267
Loops
in dependency diagrams, 304
in instance snapshots, 566
loop invariants in, 267
in static type diagrams, 569
in static type models, 568
in type diagrams, 568
Lotus, and replication technology, 443
Lower interfaces
for customization, 459–460
and infrastructure services, 460
Luckham, D. C., 712

M

Macintosh file system, and recursive composite pattern, 567
Maintenance, and components, 397
Maintenance team, and documentation, 192–193
Make a Business Model, Pattern 14.2, 530, 533, 543, 546, 548–550, 587
Make a Context Model with Use Cases, Pattern 15.5, 531, 533, 545, 587, 591–594, 596, 607, 661
makeCustomer action, in requirements spec, 432
make facility, 317
makeLink, invoking, 670, 671
makeOrder action, 354, 355, 439
in requirements spec, 432
Manna, Z., 707
Many–many associations, modeling as separate types, 561
Many–many links, 671
Many–many mapping, 586
Maps and Mapping
between abstract actions and their realizations, 166
to abstract exception action, 336
from collection elements, 689
and connecting events, 426
design action sequences to abstract action specs, 639
Maps and Mapping *(cont.)*
of invariants, 71
many–many, 586
name, 314–315
of object-oriented databases, 502
and operation specifications, 110
and recursive composite pattern, 567
string-based, 421
strings to object identities, 263
and user-interface review, 614
and user interface sketches, 524
ofuserrolesinsystemcontexttorolesinbusinessmodel, 594
Marked types, 366
Marshaling code, 402
Masking, of specs, 257–258
Master-slave listbox, 500
McConnell, S., 713
MCI Systemhouse, and UML, 697
Mellor, S., 705, 711, 712
Membership, in video store, 626–628
Membership Manager component, 682–683
Mentoring, and Catalysis adoption, 703
Mentors, assigning, 87
Message categories, in Smalltalk, 148
Message protocols, in Smalltalk, 148
Message queues, 650
Messages
and actions, 222–224
and communication components, 525
defined, 144
Message sequence diagrams, 182
Metadata, with COM+, 403
Metamodels, 371
Meter, in small components kit, 405, 406
Method class, 399
Methods
adoption of, 540
and JavaBeans, 399, 400
redefinitions of, 318
Meyer, B., 267, 706, 709, 710
Microsoft
components with COM+, 401–403
DNA project of, 396
Microsoft Transaction Server, and persistence, 392
Middleware
capabilities of, 542
communication, 496
and infrastructure components, 652

separated from business components, 650–651
and technical architecture, 520
Mikhajlova, A., 709
Military or organizational hierarchies, and recursive composite patterns, 567
Mirroring, 3
of business types in software, 646
of framework specs by collaborating components, 480
in object-oriented design, 274
and simulations, 276
Mobile code, connectors supporting, 397
Model abstractions, 216, 217, 220–222, 230
Model compositions, theoretical influences on, 710–711
Model conformance, 242–243
documenting with retrieval, 239–240
drawing, 240–241
Model diagrams, 208
Model elements
and importing packages, 294
UML, 297
Model frameworks, defined, 557
and architecture, 491
combining, 466–468
concepts summary, 378–379
for model generalizing, 564
overview of, 339–341
and template packages, 299
as templates, 13–14
theoretical influences on, 711
of types and attributes, 342–346
usefulness of, 340–341, 380, 634–637
using for video case study, 634–637
Model refinement, 221, 222
and action refinement, 250
Models and modeling. *See also* Behavior models; Business models; Frameworks; Interaction models; Static models
action, 139
"agent-based," 711
architectural, 481, 483, 491
architecture-centric approach to, 705
as-is, 624
behavior, 34
built from frameworks, 322, 448
business, 522–523
and Catalysis process, 521–522
and collections, 63–65
component-port-connector, 388, 410–412
and componentware management, 447
composing, 321, 322
of constituent components, 433–435
and design: business to code, 508–510
domain, 522–523
dynamic, 300
entity-relational (E-R), 549
feedback from, 549
formalized, 569–570
improving by defining states, 605
interactions in building, 193
interpreting for clients, 613–615
joining static, 325–326
levels of, 35–36
with objects, 706
and object states, 56–66
parameter, 101
parts of, 43, 45
problem domain/business, 35–36
relational, 549
reviewing, 599
situations for composing, 322
spiral, 513
static, 29, 34, 40, 139, 154, 216, 239, 300, 609
static type, 12
and technical architecture, 525
to-be, 624
type, 33, 581
and variability, 448
variations of subtyping in, 115–117
well-factored for documentation, 191–192
Model templates, using packages, 711
Model-View-Controller (Smalltalk), 460, 482, 498
Modifiability
and architecture, 481
and development-time qualities, 487
Modules, 386
Monitor mechanism, 496
Morgan, , 219, 267
Motor, in small components kit, 406
Mouse messages, and output events, 422
Multiple action specifications, 111–113
Multiple appearances, 324, 325

Multiple classes, inheritance from, 145
Multiple code frameworks, composing, 470
Multiple imports, and name conflicts, 296
Multiple interfaces, with components, 397
Multiple iterations, milestones with, 513
Multiple partial specifications, operation constrained by, 112–113
Multiple supertypes, 116
Multiple variants, and inheritance, 472–473
Multiplication specifications, writing, 707
Multiplicity, 524
Multiplier, in small components kit, 406
Multithreaded systems, programming, 492
Multitier peer-to-peer architecture, 516
Mural, 288, 377
Mural specification tool, 710
must-precede operator, 360, 362
Mutable types, 60
  type constant use for, 66
Mutative behavior, and equality-like relations, 368
MVC design, and changes in UI state, 270

N

Name clashes, avoiding, 366
Name conflicts
  and multiple imports, 296, 312–315
  with packages, 314–315
Named effects, 120, 129
  and business rules, 551
Names and naming
  of elements, 308
  for events, 426
  and features, 291
  and joining packages, 325
  package qualified, 699
  in packages, 285, 288, 290, 295, 318
  and placeholders, 378
Names and naming *(cont.)*
  for scenarios, 593
  in state chart, 249
  and stereotypes, 371
  substituting, 365
  of template packages, 364
Narrative
  descriptions, 514
  diagrams, 613
  exceptions incorporated into use case, 336
  joining, 331
  in package, 290, 291, 318
Narrative documentation, in packages, 288
Narrative requirements statements, 586
Narrative use case, 588
  template, 164
N-ary associations, 564
Navigability arrowheads, and class diagrams, 207
Navigation, and objects, 52
Navigation expressions, 52
  in action specs, 88
Nested packages, 285, 288, 308–309, 316, 700
  flattening of, 309–310
  generic, 364
  and imports, 309
  and joining, 325
  renaming, 315
  for semantic rules, 372
Nested states, 126
Nesting
  of packages, 699
  of state charts, 128
Network diagram symbols, for physical architecture, 485
Network Expansion, in BluePhone, 302
Network Expansion Detailed Tasks, in BluePhone, 302
Network load, 485
Network_Maintenance, in BluePhone, 303
nextProspect action, in requirements spec, 432
Node failures, 485
No-loops package, 566
Nondeterminism, 162
Nonfunctional requirements
  architecture evaluated against, 541
  and requirements activity, 518
Non-object-oriented frameworks, 465
Nonruntime quality objectives; architecture evaluated against, 525
N * 0 operator, 68
Normal defined behaviors, 332
Normalization; and separation of concepts, 560–561
Normalized data models, 705
Normal names, 333

Notations
component, 420, 452
connector, 404, 411, 452
continuity of, 198
design, 185–186
formal, 189, 190
for layered architectures, 495
and model interpretation for clients, 614
to model sets of ideas, 274
and packages, 287
for parameterized types, 700
template, 341
text-based, 288
UML pattern, 697
unambiguous, 516
for Unified Modeling Language, 705
visual, 428
Notes, in package, 290
notify message, and Observer, 677
Nouns
mapped to object types, 558
sketched as types, 552
Null, 89
in business rules, 431
Null attributes, 68
Null value, and unconnected attributes, 58
Number abstract class, 244
NumberAdapter class, 245

O

Object abstraction, 216, 217, 225–228, 230, 263
Object access, 261–263
and adapters, 263
direct, 261
and façades, 261–263
Object behavior
defined, 80
described with type specification, 139
and encapsulation in plug-ins, 678
implementation of, 590
and kinds of actions, 85–86
and modeling business and software, 83–85
and objects and actions, 80–86
pre- and postconditions specify actions, 80–82
snapshot pairs illustrating actions, 80
and types, 82–83
Object behavior models, refining, 707
Object collaborations, designing, 153–154
Object compositions, collaboration abstracting of, 181
Object conformance, 254, 263–264
documenting, 259–261
Object Constraint Language (OCL), 87. *See also* Unified Modeling Language
in Catalysis, 693
and choice of (->), 694
defining basic types using, 695–696
and enumeration types, 702
extending in Catalysis, 694–695
and flat sets, 706
summary of, 689–693
in UML, 53
Object creation, and factories, 674
Object design, 11
and distribution of objects among hosts, 651
Object Development from Scratch, Pattern 13.1, 507, 533–534, 548
Object diagrams, 418
Object equality, 45, 53
Object history, kebab or skewer model of, 89
Object identity, 45, 367
Object-instance boxes, 48
Object instance diagrams, 628
Object interaction graphs (OIGs), 182, 259
Object layer, *vs.* component layer, 30
Object Locality and Link Implementation, Pattern 16.12, 531, 534, 664, 665–666, 667
Object Management Group (OMG), and CORBA, 403, 411
Object model, designing, 257
Object-oriented databases, and architecture, 502
Object-oriented design, 26, 30–31, 39, 43, 143
benefits of, 146
business type mirrored in software with, 646
classes mirroring specification model in, 242
and collaborations, 346
vs. component-based design, 391–392
and decoupling, 643
document types in, 197
Golden Rule of, 611

implicit unique key in, 53
mirroring in, 274
vs. procedural style, 153–154
questions in, 154
and reengineering pattern, 535
subtleties in, 705
traceability of, 233, 534
and well-decoupled components, 640
Object-oriented development (OOD), 3, 165
Object-oriented frameworks, 461–465
class library with traditional reuse, 461–462
and code reuse, 461
and contrast between traditional and framework styles, 464–465
framework-style reuse and template method, 462–463
Object-oriented implementations, refinement and subtyping in, 709
Object-oriented languages, and type as object behavior, 706
Object-oriented messages, 156
Object-oriented methods
basic forms of refinement to, 709
lack of focus on user tasks in, 705
Object-oriented programs, 6
benefits of, 277
building, 508
kinds of variables in, 144
Object-oriented programming, 114, 264, 265, 384
and business stimulus, 545
and collaboration design, 347
origins of, 276
strong decoupling in, 265–266
Object-oriented programming languages, 7, 143
and inheritance, 474
messages in, 159
and plug-ins, 459
Object-oriented programs, speed of, 667
Object-oriented systems, architectural issues with, 712
Objectory, 370
Object refinements
with adapter objects, 263, 264
boundary decisions in, 263
essence of, 259
process of, 256–259
spreadsheet, 254–264
Object request broker (ORB), 540, 650, 663
Objects, 3, 7, 45
abstracting together with actions, 227
actions affecting, 4–5, 40
actions localized to constituent, 258
actions representing, 251
attribute values of, 51
behavior of, 216
and collaborations, 216, 347
and components, 392
vs. components, 390
constituents, 16, 20, 21
copying, 369
creating, 146
defined, 49–50, 84
described by types, 33, 57–58
designing in collaborating groups, 108
at different scales, 9–10
distinguished from types, 552
documenting, 556
and equality, 366
factory, 31
finding, 549
and flat files, 501
and generalization, 672
grouped into types, 82
head, 493
identity of, 53–54
implementing, 658
interactions between, 139, 167
interactions between members of groups of, 178
joint behavior of, 181, 182
and links, 664
in modeling development process, 521
modeling with, 43–44, 706
modeled in documentation, 191
and navigation expressions, 52
newly created, 90
polymorphic coupling between, 473
principal, 478
role-playing, 358
roles of in frameworks, 635
separating within frameworks, 469
separation of static, dynamic, and interactive modeling of, 706
in snapshots, 48

symbols for different, 53
symbols for same, 53
systems and subsystems as, 712
in type specifications, 328
variables distinguished from, 144
what they mean and don't mean, 255–256
zoomed-in, 10
Object state, and objects and attributes, 49–54
Object state implementations, 54–56
business world implementations, 56
Java implementations, 54–55
other implementations, 56
relational database implementations, 55
Object technology, 274, 651, 701
benefits of, 275
change handling by, 539
and granularity of components, 394
and reusability, 446
Object types, 57
and architecture, 493
definitions for, 75
nouns mapped to, 558
in video store case study, 571
Object type specifications, for calendar, 102
Observation and interviewing, 549, 552
Observation framework, 340, 351, 352
with Two-Way Link framework, 353
Observation pattern, 340
Observer class, 676
Observer, Pattern 16.17, 178, 676–677
OCL. *See* Object Constraint Language
OclAny object, in OCL, 692
OclType (a metatype), in OCL, 692
ODBC, 410
ODMG, 502
OIGs. *See* Object interaction graphs
ok()function, 71
OLE/COM (Microsoft), 384, 402
OMG
and UML, 697
vertical standards in, 396
OMG Web site, UML reference material on, 705
OMT book (Rumbaugh), 705
O * n operator, 68
1-N association, 500
One-to-one links, 671
OOD. *See* Object-oriented development
OO Golden Rule (Seamlessness or Continuity), 273, 274–275
OORAM
contributions of, 705, 708
role-model synthesis, 710
Open arrows, for events, 423
Open-Closed Principle, and reuse economics, 455
Open collaboration, 173, 179
and designing joint service, 177–178
Open object systems, interaction protocols for, 709
Open systems design, 149
Operating systems, and platform independence, 649
Operation abstractions, 216, 217, 218–219, 225, 230, 268
what it means and doesn't mean, 264–265
Operation conformance, 268
vs. action conformance, 268, 269
Operation invocation sequences, 702
Operation refinement, 219
to component-level action, 259
spreadsheet, 264–268
Operations, 82, 85
on collections, 65
constrained by multiple specifications, 112
defined, 144, 264
descriptions by, 167
differences between effects and, 709
editor's, 131
implementing, 332
merging, 323–324
on sets in abstract descriptions, 573
specifications and outcome of, 110
and variables, 144
Operation spec conformance, documenting, 266–268
Operation specifications, 110, 151. *See also* Type specifications
combining in Catalysis, 697
comparing, 266–268
factoring in component specifications, 203–204
and object's behavior, 216
precision of, 100
simplifying, 102
and system specifications, 519
Operators
Boolean, 68
Collection, 69, 91

Optimization, 654
  for performance, 277
Optimization, Pattern 16.13, 531, 534, 667–668
"Optional" marks, 663
ORB. *See* Object request broker
Organizational maturity, 32
Organization charting, 700
Orthogonal Abstractions and Refinement, 273, 278–279
oscustomer link, 437
Outages, avoiding, 354
Outgoing interfaces, in COM, 402
Out parameters, 134–135
Output events, 426
  specifying, 421, 422
Output parameters, types required for, 124
Output ports, 415
outputPort.sinks, 425
Output properties
  input properties driven by, 425
  specifying, 423
Outputs, 161
  constraints applied to, 425
  inputs and, 161–162
  and system actions, 600
Outputs of actions, 134–137
  and out parameters, 134–135
  raised actions, 135
  and return values, 134
  and sequence expressions, 136–137
  specifying sequences of raised actions, 135–136
Outside
  and projects, 526
  and requirements activity, 518
  separated from boundary and inside, 526, 528
Oval class
  in Class library with traditional reuse, 462
  with framework approach to reuse, 463
Overloaded functions, attributes written as, 325

P

Package diagrams, 652
  with renaming, 315
  symbol, 287
Package imports, 697
  and class inheritance, 293
  structure of, 292, 293
Package nesting, and package provisions, 364
Packages, 211, 318–319
  architecture, 416, 495
  and build, 317
  built upon imports, 321
  business activities modeled with, 299–301
  Catalysis, 378, 515, 694
  change management and upgrades in, 515
  component, 387, 445
  concrete, 297, 298
  container, 700
  contents of, 287–289, 316
  coupling between, 295
  decoupling with, 303–307
  defined, 18, 285–286, 287
  definitions and joins in, 324
  and dependencies, 96, 319
  to distinguish components, 590
  documenting import structure of, 525
  encapsulation with, 310–312
  explicit parameters for, 366
  extending, 292
  for grouping classes, 710
  hiding, 315
  and horizontal slices, 301–303
  imported by other packages, 286, 337
  imports and importing of, 292–298, 319
  interpreting contents of, 370–371
  and iterations and increments, 513
  joining, 325
  model of, 290
  names for, 308
  and narrative and dictionary, 290–292
  nested, 288, 308–309, 315, 316
  no-loops, 566
  and notation, 287
  and parallel work, 541
  programming language, 318
  publication, version control, and builds, 315–317
  purpose of, 319
  and redefinition, 318
  refactoring across, 642
  refactoring contents of, 524

in requirements specifications models, 18
rules for system architecture in, 510
and selective hiding, 315
semantics for, 369–373
to separate design units, 34
separation of concerns in, 298–299
structure and intended content of, 526
and system specifications, 519
template, 299, 564
theoretical influences on use of, 710
typical structure of, 526–530, 527
unfolded forms in, 337
Unified Modeling Language, 699–700
using, 285–319, 298–303
and vertical slices, 299–301
viewpoint, 300
virtual, 297, 298, 377
Package structure, documentation structured around, 708
paid Session, 126
Panels, in user-interface components, 525
Parallel Work, Pattern 13.4, 541–542
Parameterization, substitution as, 365
Parameterized attributes, 45, 59–61, 68, 697, 706
derived, 7
and joining, 325
and models of state, 698
and states, 603
vs. read-only operations, 119
Parameterized states, 605
Parameterized types, notation for, 700
Parameter models, creating, 101
Parameters, 144
of action, 160
clarifying meaning of, 708
definitions for, 75
distinguished from parameter types, 552
for system behavior specs, 597
Parameter types, 111
parameters distinguished from, 552
Parent packages, 308
Partial abstract classes, 304
Partial definitions, and joined classes, 331
Partial specifications, writing, 112
Partitioning
of model between components, 21–23
and pluggable components, 29
subtype, 115
type, 116
Parts (Digitalk), 408
Pattern Languages of Programming (PloP) design series, 712
Pattern reuse, 41
Patterns
books on modeling, 713
component connectors in, 712
and componentware management, 447
defined, 341
design of, 698
detailed design, 669
domain-specific modeling, 698
Enterprise JavaBeans, 494
event notification design, 493–494
in features, 291
in interface packages, 494
and package content partitioning, 299
in packages, 288
subsystem controller, 494
in user interface, 500
Pattern symbol (UML), 344
Peer classes, façades for, 649
Peers, and façades, 263
pendingInvoice Session, 126
Performance, 237
optimization of, 277, 667
and partitioning components, 639
and runtime qualities, 489
and system architecture, 503
Performance constraints, 276
Performance improvements, 275
Performance monitoring, 300
Performance requirements, of use case, 164
Persistence, 384
and business objects, 501
and components, 392
with Java, 400
and object-oriented databases, 502
and platform independence, 649
Persistence domain, 299
Persistent storage, for components, 391
Petri Net interpreter, 278
Phased development, 32

Phone_Basics package, fundamentals in, 303
Physical architecture, 485
Pictorial models, 188
  specifications written as, 246
Pictorial notation, disadvantages of, 371
Pictorial representation. *See also* Diagrams
  in business model glossary, 558, 559
Pictorial statements, in packages, 289
Pictorial type expressions, common, 115–117
Pictures, in narrative of package, 291
Pipe-and-filter architecture (UNIX), 394, 493
Placeholder actions, and pluggable roles, 480
Placeholders (), 342, 355, 378, 448
  and arguments, 374
  in frameworks, 179
  names written as, 13
  substitutions for, 362
  of template, 364
Platform independence, 651
Platform Independence, Pattern 16.5, 549
Platform or architectural constraints, and requirements activity, 518
PLoP. *See* Pattern Languages of Programming design series
Plug conformance, and components, 449
Pluggability
  of components, 323, 388, 410
  and parts, 37, 39, 40, 385
  and roles, 478, 480
Pluggable design, theoretical influences on, 712
Pluggable reuse
  and components, 395
Plug-ins, 463, 472
  implementation mechanisms for, 27–28, 471
  and plug-points, 458, 678–679
  sameAs(x) query for, 478
Plug-points, 453, 463
  class-based frameworks with, 524
  factoring, 465
  in generic components, 459
  implementation mechanisms for, 471
  for plug-ins, 458
Plug-Points and Plug-Ins. Pattern 16.18, 675, 678
Plug technology
  basic, 471–477
  and good uses for inheritance, 474
  inheritance and template method, 472–473
  inheritance replaced with type-based composition, 475–476
  polymorphism and forwarding, 473
  specifying super/sub interface, 476–477
  templates, 471
"Plug-togethers," 678
Pointers
  attributes and associations into, 207
  and two-way links, 670
"Policy" plug-ins, 678
Polymorphism, 39, 143, 145, 146, 214, 276
  and forwarding, 473
Port
  attributes, 425
  in Catalysis, 414
Portability
  and architecture, 481
  and development-time qualities, 488
  and system architecture, 503
Port categories
  in Catalysis, 414
  in Cat One, 415
Port.component, 425
Ports and porting, 649
  and components, 388
  connectors between, 411
  defined, 410
  and façades, 263
Positive and negative examples, reviewing, 613
POSIX interface, 460
Post, 5
Postconditions, 4, 6, 12, 27, 28, 40, 139, 141, 370. *See also* Preconditions
  action, 13
  and action signatures, 326
  actions specified by, 80
  advantages with, 23
  anding, 326
  checking, 250
  and collaboration abstracts, 181
  in collaboration frameworks, 348, 349
  and concurrent actions, 169
  connections specified in, 428
  designers working from, 603
  of events, 423
  explicit, 111
  and external actions, 179

factored by effects, 119–120
formalizing, 88
invariants absorbed into, 253
and joined classes, 331
and joint actions, 701
meaning of, 171–172
more precise, 92
and newly created objects, 90
operation defined by, 119
of operations on object, 220
and operation specifications, 151
with output events, 426
and parameter types, 111
and partial specifications, 112
and quoted actions, 122
referring to other actions in, 122
and requirements management, 586
retrieval of, 23
and rigor, 514
snapshots for guiding, 87–88
special terms in, 142
in specifications, 195
and system actions, 600, 602
testing, 267
and type intersection, 324
and variable degree of rigor, 32
writing, 86–87
Practicality, compromise with, 274
@pre, 6, 89, 136, 370. *See also* Effects; Postconditions
applying, 88
and effects, 119
postconditions containing, 267
in use cases, 573
Pre- and postcondition pairs, ANDing of, 266
Precise abstractions, 89, 214
Precise action specifications, 86–92
comparing before and after, 88–90
snapshots used to guide postconditions, 87–88
Precise specifications
and collections, 90–91
more precise postconditions, 92
newly created objects, 90
and preconditions, 91–92
and variables action specs uses, 90
Precision, 37, 38–39, 41, 68–69
abstraction without, 223
and business rules, 551
defined, 38
and preconditions, 92
and safety-critical projects, 515
with type specifications, 97, 599
Precondition/postcondition pairs, effect specified as, 79
Preconditions, 139, 141, 265, 370. *See also* Postconditions
and action design, 656
and action signatures, 326
actions specified by, 80
anding, 326
and assertions, 267
and concurrent actions, 169
and debugging, 151
“design-time,” 364
explicit, 111
invariants absorbed into, 253
and joined classes, 331
and localized actions, 159
and modeling for video store case study, 574–575
operation defined by, 119
and parameter types, 111
and partial specifications, 112
and precise specifications, 91–92
and rigor, 514
and system actions, 600, 602
testing, 267
and transitions, 604
and type intersections, 324
Preconditions and modeling, for video case study, 573–574
Predefined context design, 621
in storyboards for video case study, 620
Predicate calculus, 699
Predicates, 377
Pre (==>) post *vs.* pre & post, 120–121
Pre/post specifications, 264
comparing, 266–267
Presentation, and infrastructure components, 652
^pressed requirement, 422
Primary problem domain, 299
Primary use case, 164
Primitive elements, in architecture, 491
Primitives and values, 699

Primitive types, template packages for, 376–377
Principal objects, 478
Priority attributes, and packages, 288
Private event, in component, 204
Private inheritance, in C++, 150
Private usage, import for, 294–295
Problem domain
  and code, 273, 275
  and modeling strategy, 283
  and orthogonal abstraction, 278, 279
  and technical architecture, 497
Problem domain models, 35, 275
  and frameworks approach, 477
Problem domain objects, 396
  roles in frameworks, 468
Procedural style, *vs.* object-oriented design, 153–154
Process and activity flow and decomposition, 700
Process architecture, 486
Process guidelines, 526
Process model, questions supported by, 522
Process patterns, 31
  development context: defining routes, 530
  object development from scratch, 533–534
  parallel work, 541–542
  phases in, 530–532
  reengineering, 535–538
  short-cycle development, 539–540
Process patterns for refinement, 273–284
  Golden Rule *vs.* Other Optimizations, 273, 276–277
  OO Golden Rule (Seamlessness or Continuity), 273, 274–275
  Orthogonal Abstractions and Refinement, 273, 278–279
  Recursive Refinement, 273, 283–284
  Refinement Is a Relation and Not a Sequence, 273, 280–282
Product development, 457. *See also* Development
Product families, and packages, 317
Product items, framework for, 635
Product life, lengthening, 583
Program code
  as action implementation, 329
  classes and associations translated to, 659
Programmers, and detailed design patterns, 669
Programming
  by adaptation, 446
  aspect-oriented, 711
  subject-oriented, 331
Programming language parameters, *vs.* action parameters, 160
Programming languages. *See also individual languages*
  and abstract classes, 146–147
  classes and types, 143–151
  class extension, 145–146
  class objects, 149
  generic types, 148–149
  internal variables and messages, 144–145
  messages and operations, 143–144
  package-like constructs in, 285
  packages for, 318
  and platform independence, 649
  and recursive composite pattern, 566
  specifications in classes, 149–151
  types of, 147–148
  types and classes in, 708
Progressive formalization
  and case study, 528
  defined, 528
Project and planning constraints, and requirements activity, 518
Project glossary, 48, 558
Project planning activity, 526
Project risks, sources of, 497
Projects
  development standards for, 525–526
  factors in failures of, 505
  funding for, 513
  highest risks in, 513
  typical evolution of, 522–526
Properties, 412, 413, 416
  bidirectional, 425
  connector, 426
  and JavaBeans, 399
  and require condition, 424
  specifying, 423–425
  templates as packages of, 359–366
  of two components, 404
Property connector, for Cat One, 418–420
PropertyConnector framework, application of, 418
propertyPort.constraint (value), 425
propertyPort.value, 425
Property values, 424

Protoclasses, 283
Prototypes and prototyping, 281, 282
  building, 616
  feedback from, 549
  in short-cycle development, 615
  and storyboards, 595
  in storyboards for video case study, 620
Provided interfaces
  in component packages, 387
  explicit, 388
Provisions, 362
  applying templates with, 363
  defined, 364
  in frameworks, 379
  in templates, 360–364
Proxy(ies), 178, 668
  code, 402
  and legacy components, 450, 451
  objects, 258
Publication, and packages, 315–316
Public inheritance, 150
  in C++, 150
Published package, 316
purchaseCourse action, 86
Pure abstract classes (C++), 147, 219, 304, 672
purge action, 626
  for video store, 629

Q

QualifiedFor(Course) attribute, 71
qualified For link, 67
Qualifiers, defined by templates, 370
Quality, and reuse, 454
Quality assurance, 526
  in Catalysis, 514
Quality assurance departments, and operation specs, 219
Query, parameterized attribute as, 59
QueryInterface, 402
Query languages, 501
Query processor, 496
Quoted actions, defined, 122
Quoting
  of action specs, 707
  of effects, 168
Quoting syntax ([[...]]), 122

R

RAD. *See* Rapid application development
RAD-style timeboxing, 188, 194
Raised actions, 135
  specifying sequences of, 135–136
Rapid application development, 31
  discussions about, 713
Rational Software, and UML, 697
Rational Unified Process, 702
Read-only functions, 239
Read-only operations, parameterized attributes *vs.*, 119
Reality. *See also* Reification
  truth and, 274
Realization, 34
  and abstraction, 157
Real numbers, in OCL, 692
Real-time control software, 276
reassign_course action, 91
Receiver
  defined, 144
  in operation invocation, 162
[[receiver.action]], 129
[[->receiver.action]], 129
Recipe, 590
Reciprocal pointers, 670
Recoverable objects, and transaction servers, 525
Recursive Composite, Pattern 14.11, 566–567
Recursive decomposition, 509
Recursive Decomposition: Divide and Conquer, Pattern 15.4, 582, 589–590, 607, 608, 609
Recursive Refinement, 273, 283–284
Recursive type diagram, 566
Recursive use-case driven design, reifying, 644–645
Redefinition and packages, 318
Redundant links and associations, 560
Redundant relationships and views, 199
Redundant specifications, 294
  defined, 105
  usefulness of, 105
Redundant terminology, 558
Reengineering

and abstraction and refinement, 537
and deployment, 533
and transition plan, 536
Reengineering pattern, 507, 535–538, 545, 547
Reenskaug, T., 705, 708, 710
Reentrant code, 458
Refactoring
of design, 521
and project evolution, 524
Refined model, for video business, 576–577
Refinement, 32, 33, 136, 200, 211, 272, 516, 703, 708, 709. *See also* Abstractions
of abstraction models, 546
action and model, 250–252
of actions, 40, 166, 167, 224–225, 661
and action types, 155–156
advantages to understanding, 238
and aggregation, 231–233
for architectural models, 484
of associations, 9
and boundary decisions in object, 263
in business (domain) model, 528
Refinement *(cont.)*
and case study, 528
in Catalysis, 639
and choosing level of abstraction, 556
of collaboration, 23, 252
and composition, 229
and conformance, 213
and constraining design decisions, 483
and context models, 593
defined, 34, 215
and description in case study, 528
and documentation between design and specification, 209
documenting, 157, 230–233, 249, 276, 355–357, 709
and essence of object, 259
exception paths defined via, 700
of formalized use case specs, 578–579
fractal process of, 255
of frameworks, 352–357
and granularity, 515
and justification and test, 510
kinds of, 34
layers of abstraction with, 34
localized, 531
model, 221, 222
in modeling development process, 521
of object behavior models, 707
of objects, 40, 256–259
of objects and actions at different scales, 6–10
in packages, 288
and plug conformance, 449
process patterns for, 273–284
recursive, 273, 283–284
reviewing, 268
and rigor, 514
and signature of abstract action specification, 335
of specifications, 611–612
spreadsheet: example, 233–238
of state charts, 269–272
and state mapping, 22
and subtypes, 231
successive, 27
for system behavior specs, 596
and technical architecture model, 525
theoretical influences on, 709
and traceability, 233, 510
and use cases, 713
of video business use case, 575–579
Refinement Is a Relation, Not a Sequence, Pattern 6.4, 273, 280–282, 664
Refinement relations, in packages, 286
Refinement symbol, 231
Refinement trees, 216–218
Reflection, 562
and building components with Java, 398–399
and JavaBeans, 398
Reflective techniques, 421
register action, 352
Registering and Registration(s)
connection established by, 417
and framework building, 352
messages, 417
Reification
of abstractions, 229
of actions, 251–252, 580
of associations, 561
and high-level component design, 643
of model types to classes, 521

Reifying Major Concurrent Use Cases, Pattern 16.3, 644–645
Reil, A., 713
Reiteration, of design types, 659
Related actions. *See* Collaborations
Relational databases, 650
  advantages and disadvantages of, 501–502
  and architecture, 501–502
  implementation of, 55
  and links between objects, 55
Relational models, 549
Relational technology, 501, 502
Relation detail, and model frameworks, 634
Release version, package-level, 316
Reliability, and runtime qualities, 489
Rely and guarantee, 169
Rely clauses, and process architecture, 486
rely conditions, 169, 264, 265
  anding of, 326
  and granularity, 170
Rely/guarantee specifications, 264
Remote access, and container, 401
Remote interface, in EJB component, 401
Remote object references, 402
removeEvent operation, 93, 94, 97, 98, 99, 100
rental invariant, in video store case study, 578
Rental type model, in video store case study, 628–629
Rental Manager component, 679–682
rent definition, in video store case study, 572
rent use case, described, 574–575
Replication, 366
  and federated architecture, 442, 443
Replicators, and architecture, 481
Represent Business Vocabulary and Rules, Pattern 14.3, 551–553
Request broker, and horizontal standards, 396
Require condition, 424
Required exceptions, *vs.* undefined behavior, 332
Required interfaces
  in component package, 387
  explicit, 388
Requirements
  abstractions, 37
  in packages, 288
  project risks and unclear, 513
  and short-cycle development, 540
Requirements activity
  primary deliverables of, 518
  in typical business systems, 518–519
Requirements and analysis, and outside and boundary, 517
Requirements documents, 586
  and clients, 193–194
Requirements elicitations, 202
Requirements management, 586
Requirements models
  decoupling in, 306–307
  and heterogenous components, 428
Requirements specifications, 230
  building, 591
  business rules in, 429–431
  described, 10
  models for, 16–18
  requirements model in, 429–430
  target operations and system context in, 432
Reservation type model, in video store case study, 630
Reservation Manager component, 681
Reservations
  generalized, 635, 636
  more-detailed, 636–637
Reservations manager, 650
reserve action, in video store case study, 576, 630
ResourceAllocator and Resource classes, 471
Resource pooling, and container, 401
Response time, 520
Responsibilities
  assigning, 25–29, 40, 658
  for classes, 659
  among collaborating objects, 347
  and collaborations, 661–663
  documenting, 592, 662
  and scenarios, 617–618
  separation of in design, 333
re_stock operation, 157
Result object, 90
Retailer/wholesaler interaction sequence, 162–164
Retrieval, documenting model conformance with, 239–240
Retrieval diagrams, 241
Retrieval functions (or abstraction functions)
  defined, 215

value of, 58
and verification, 221
Retrieval mapping, 511
Retrievals
in business models, 437–439, 443
of distributed architecture to specifications, 685, 686
«retrieval» tags, 240
Retrieve functions, 150
and bug discovery, 150
and operation specs, 266
Return, 25
return action, 600
for video store case study, 577, 629, 630, 631
return type, 134
return use case, in video store case study, 579
Return value, 134, 135, 159, 335
and decoupling, 265
Reusable artifacts, defined, 455
Reusable design, and packaging, 285
Reuse and reusability
and adoption of object technology, 446
Reuse and reusability *(cont.)*
of assets, 457
and components, 397, 444
components and pluggable, 395
considerations for, 535
constraints on, 276
and decoupling, 641
and development process, 453–457
and development-time qualities, 487–489
improvements in, 278
and inheritance, 472
and model frameworks, 380, 448
and packages, 318
and partitioning components, 639
and theoretical influences on pluggable design, 712
traditional approach to, 461–462
and user interface, 500
Reuse culture, 456–457
Reuse Law I, 455
Reuse Lemmas, 455–456
Reverse-engineering
and development evolution, 525
and user manuals, 549
Rigor
and modeling Catalysis process, 521
"on-demand," 514
RMI, 649
Role-activity diagrams, 713
Role-centric modeling, and OORAM approach, 705
Role decoupling, 265, 676
Role Decoupling, Pattern 16.15, 660, 672–673, 676, 678
Role delegation, and plugging together code components, 478–479
Role models, and OORAM method, 708
Role objects, building objects by connecting, 478, 479
Roles
within frameworks, 635
interface defined for, 474
pluggable, 480
shared state observed by, 480
uniform composing of, 470
Role separation, with context models, 593–594
ROOM, basis of, 712
Root type, 137, 138
Routes, development context and defining, 530
RTTI. *See* Runtime type identification
Rules/guidelines, in architecture, 409
Rumbaugh, J., 705
Rummler, G. A., 713
Run, and user interfaces, 498, 500
run_course, 8
Runtime environment, and platform independence, 629
Runtime patterns, and architecture, 515
Runtime qualities, 490
and architecture, 481, 488–490, 525
Runtime type identification (RTTI), 149

S

Safety-critical projects, and precision, 515
Saini, A., 711
Sales Client component, 441, 442
salesOrder action, 440
SameAs(x) query, for plug-ins, 478
Sandbox model, 316
Scalability
and components, 398

and federation, 407, 443
and inheritance, 472–473
large-scale, 652
with multitier design, 400
of object-oriented databases, 502
of relational databases, 502
and runtime qualities, 489
and system architecture, 503
Scales, objects and actions at differing, 6–10
Scenario-based evaluation, 491
Scenario diagrams, 182
and action occurrence, 176
sequence, 176, 177
Scenarios, 200, 202, 614, 708
and architecture evaluations, 481, 490–491
and business model, 519
and collaboration uses, 177
and concrete reviews, 613
feedback from, 549
rental with new member creation, 618
snapshots for, 600
and system behavior specs, 599
and system context diagrams, 518
and technical architecture models, 525
and use cases, 593
and usser interface sketches, 616–618
user interface *versus* system operations, 624–626
for video case study, 615–619
for video rental, 617
and workshops, 205, 206
schedule_confirmed_course, 120
schedule_course action, 84, 90, 104, 110, 119 , 120
scheduleCourse operation, 86
Schedules attributes, and packages, 288
schedule_unconfirmed_course, 120
Scoping, 33
Screen position, and object identity, 263
Screen-scrapers/screen-scraping, 392, 393, 651
Scripts, 341
Scrollbars, in user-interface components, 525
Seamlessness (or continuity), 274
Secondary use case, 164
Security
with COM+, 403
and container, 401
and Enterprise JavaBeans tier, 498
and horizontal standards, 396
and runtime qualities, 489, 490
selected_product attribute, 157
Selective hiding, 315
Selector, in small components kit, 406
select(self), 131
Self-contradictory models, in packages, 313
Selic, B., 705, 712
sell definition, in video store case study, 572
sell use case action, 571
revised and improved business collaboration for, 573
Semantic relationships, in Catalysis models, 515
Semantics, 370
layered, 377
package, 369–373
rules for dialect, 372–373
Semaphone primitive, 493
Semicolons, statements separated by, 267
Seminar business application, orthogonal abstractions and refinement with, 278
Seminar System, roles of, 15–16
Seminar System Implementations, 27
SeminarSystem::Instructor, 21–23
Sender, in operation invocation, 162
Sensible-sequences state chart, 181
Separate Middleware from Business Components, Patterns 16.6, 531, 543, 650–651
Separating Façades, Pattern 16.4, 646–648
Separation of Concepts: Normalization, Pattern 14.8, 560–561
Separation of concerns, and packages, 298–299
Sequence charts, 155
Sequence constraints, and collaboration specifications, 180–181
Sequence diagrams, 8, 15, 20, 261
and action abstractions, 222, 223
with actions, 176–177
for concurrent actions, 170, 171
in documentation, 190
and scenarios, 177
Sequence expressions, defined, 137
Sequences, in collection types, 689, 691
Servelet, 498
Server components, container for, 401
Servers
in COM, 402

and components, 388
ServiceEngagement type, 113, 114, 115
SessionGlue object, 469
SessionJob class, 469
Sessions, and communication components, 525
SetAddition, performing, 250
set methods, 482, 483
Sets, 376, 377
and Catalysis, 699
in collection types, 689, 691
flat, 91
operations on, in abstract descriptions, 573
types treated as, 693
setSum action, 235, 248, 270
and refinement of state charts, 269–270
SetX(x) method, and architecture, 482
S gate, in component for Cat One, 427
Shaded bubbles, in action sequence diagram, 176
Shape class
with framework approach to reuse, 463
with traditional reuse approach, 461
Shared objects, with many interfaces, 469
sharedShapesEq operator, 369
Shaw, M., 712
Shlaer, S., 705, 711, 712
Shop object, 358
Short-cycle development
cycles of, 31
and users' feedback, 615
Short-Cycle Development, Pattern 13.3, 507, 539–540, 541
Short-life products, 187
Shutdown mechanism, 496
Sibling packages, 308, 309
Signatures, 141, 326
of abstract action specifications, 335
in action description, 169
of operations of component, 203, 204
"Significant" changes, 424
Silent transitions, 133
Similar equality, 367
Similarity relationships, defining, 53
Simula language, 274, 276
Simulations, and mirroring, 276
Single key quality, and architecture, 490
Single type specifications, 18
SinkPort, on component diagrams, 418
Sinks
links, 418
registry of, 417
and Transfer ports, 416
SinkType, in component diagram, 419
Slide shows, 595
Small components, kit of, 405–406
Smaller-grain components, 391
Smalltalk, 472
block in, 699
class objects in, 149
class representation in, 562
components in, 387
and containment, 145
Envy, 318, 710
inheritance support with, 474
message categories/message protocols in, 148
message names in, 415
packaging forms in, 318
types in, 147
Snapshot pairs
actions illustrated by, 80
and action specifications, 108, 109
interpretations by, 613
Snapshots, 13, 40, 54, 57, 614. *See also* Diagrams
for action occurrence, 98
actions specified with help of, 531
allowed in static models, 48
alternative ways of drawing, 50–52
to animate specs for spreadsheets, 234–235
for architectural models, 484
and associations, 62
and attributes, 50
attribute types and legality of, 58
and basic design, 657
in business models, 202
and concrete reviews, 613
and diagrams, 45
drawing pictures of states with, 47
generalized by type models, 108
and glossary, 559
to guide postconditions, 87–88
to illustrate actions, 706
instance, 566
and intent of model, 553

limitations with, 80
of membership actions, 627, 628
object instances in, 65
and sequence diagrams, 177
of software system states, 17
spreadsheet, 236
and static invariants, 66–67
and static models, 77
and system action specifications, 600, 601
and system behavior specifications, 596
for system requirements specifications, 17
for verification, 550
Sockets, 27, 28
Software
action conformance within, 253–254
architecture, 712, 713
built from components, 322
business types mirrored in, 646
Software component context diagrams, 229
Software design
and conformance to spec, 640
refinement and, 252
Software development, cycles in, 540
Software development projects, documents in, 196–197
Software distribution, and architectural views, 485
Software implementation, and horizontal packages, 301
Software installation, 526
Software location, and business location, 651
Software platforms, and technical architecture, 520
Software productivity, 384
Software requirements, and business rules, 551
Software specifications, and horizontal packages, 301
Software systems, and business models, 624
sold Session, 126
Solid arrows, for input and output properties, 423
Sorted List frameworks, 361
substitution provisions in, 363
SortedList object, 262
Sorted lists, 360, 361, 362
Sorted List Template, 361, 362
Sorted queues, 268
Source code
in packages, 316
and reusable artifacts, 455
source links, 418
SourcePort, on component diagram, 418
SourceType, in component diagrams 419
Specifications. *See also* Implementations
fulfilling of, 266
and implementation and testing, 510
of object behavior, 509
and objectives in designing to meet, 639–640
and object refinement, 256–257
recursive across business or domain model, 507–508
snapshots in, 248
Specification cycle, requirements and, 586
Specification frameworks, for seminar business, 466–468
Specification functions, 272
Specifications, 586
action exceptions and composing of, 331–337
for actions, 256
in class diagrams, 260
in code, 243–246
collaboration, 179–182
component, 505
convenience attributes and simplifying of, 118–119
convenience state modeling elements for simplifying, 707
derived, 105
directly translated to code, 244
and encapsulation, 311–312
extension of, 115
fulfilling, 266
and kits, 410
masking, 257
and model frameworks, 339
necessary and unnecessary parts of, 104–105
in packages, 289
and plug conformance, 449
port attributes used in, 425
of raised actions, 135–136
redundant, 105, 294
refining, 611
reuse of, 455
“stubbing” of, 526
Specifications actions, and business goals, 247
Specifications documents, defined and discussed, 195, 196

Specification types, 174
  classes, design types and, 654–656
  vs. design types, 123–124
  factoring to, 124
  state charts of, 130–131
  system action specs factored into, 601
Specify Components, Pattern 15.1, 581–582, 583–584
Specify Component Views, Pattern 15.10, 587, 607–608, 609
Specify-implement-test cycle, 541
Specifying a System Action, Pattern 15.8, 531, 582, 600–602
Spending, in program's life cycle, 276
Spiral models, and design, 513
Spivey, J. M., 706, 710
Split, in component for Cat One, 427
Spreadsheets
  and application architecture, 520, 525
  cells in, 676
  implementations for, 236–237
  model refinement, 238–246
  object refinement, 254–264
  operation refinement, 264–268
  refinement example, 233–238
  specifications for, 234–235
SpreadSheet abstract class, 244
SpreadSheetSpec package, 244
SQL, 55, 393, 651
  querying using, 501
Square brackets, as guards, 576
squareRoot, 110, 111
Stack action, 327
Stake-holder roles, 526
Stakeholders
  and architectural system qualities, 481
  concerns and requirements of, 486–487
  and quality attributes, 490
  reducing misunderstandings between, 583
Standards and standardization
  for component architecture, 412
  and components, 395–397
  connector, 397
  creating common, 554
  horizontal, 396
  leveraging, 398
  vertical, 396
Standard Template Library, 711
Standish Group, 505
Stapleton, J., 713
Start-up mechanism, 496
State abstractions (attributes), *vs.* invocable operations, 119
State-chart-centric behavior models, and Syntropy, 705
State charts, 126–133, 157, 180, 181, 252, 552
  and action conformance, 249, 254
  and action sequence refinements, 604
  action specs, snapshots and, 707
  and ancillary tables, 133
  behavior described in, 519
  and behavior of system, 633
  in component specifications, 204
  copy, 633
  defined, 130
  elements of notation for, 128
  indeterminate, 132
  nested, 128
  refinement of, 269–272, 709
  for sequence expressions, 137
  and silent transitions, 133
  of specification types, 130–131, 531, 598–599
  as specification vehicle with objects, 707
  state in, 126, 603
  and states as attributes and invariants, 126–127
  and state transitions as actions, 127–129
  and storyboards, 595
  translating state transitions to actions, 129–130
  and undetermined transitions, 131–132
  used in system type models, 603–606
  for video store case study, 633
State definition matrix, 133, 134
State diagrams, 270
  validity of, 271
State-event matrix, 606
State-machine interpreter, 496
State models, and technical architecture models, 525
States. *See also* Effects
  action occurrences and changes in, 81
  for architectural models, 484
  and attributes, 269
  choosing, 603
  comparison between, 91
  defined, 127, 522, 578, 605

describing, 76
invariants expressing constraints on, 77
nested, 126
and object's behavior, 46, 47
and observer pattern, 676
output event and change of, 422
pictorial representation of, 576
placeholders and changes in, 480
and snapshots, 47
and system actions, 600
transitions between, 127
and type models, 598
State-specific constraints, 73
State transition diagrams, 498
State transition matrix, 133
State transitions, 370
defined, 130
and effects, 167
translating to actions, 129–130
State types, 116–117, 138
and behavior models, 116–117
used for rental in video store case study, 578
Static associations, in business models, 548
Static dependencies, and architecture, 515
Static invariants, 45, 66–73, 107, 117, 568, 702
and behavior models, 106
and business rules, 551
in collaboration specifications, 180
defined, 67
parameterized attributes constrained by, 119
and snapshots, 613
testing, 514
and time-triggered actions, 594
and use case, 165
Static models, 17, 29, 34, 40, 43, 139, 154, 239, 300
for business and components, 75–76
for components, 609
defined, 73
described, 46–49
and invariants, 12–13
main purpose of, 77
object attributes and invariants of, 706
object attributes and variants of, 45–77
and object's behavior, 216
reasons for making, 80
and snapshots, 48, 77
theoretical influences on, 706
using, 48–49
Static object-oriented models, *vs*. entity relational models, 560
Static type diagrams, 98, 99
loops in, 569
Static type models, 12
in business modeling, 202
for component specifications, 205, 208
and invariants from association loops pattern, 568
Status attributes, and packages, 288
Stereotypes, 371, 377, 378
for architectural categories, 495
defined, 372
and dialects, 371–372
and physical architecture, 485
in Unified Modeling Language, 698
STL, and framework provisions, 711
Storage elements/mediums, 491
and object locality, 665
Storyboards, 194, 205, 208, 514
and business models, 518
and use cases, 593
and user interface sketches and flow of windows, 622–623
for video case study, 619–621
Storyboards, Pattern 15.6, 582, 595
Strategy objects, 473
Streams, connectors supporting, 397
String-based mapping, 421
Strings, to object identities, 263
Stub code, 402
Stub operation specifications, 657
Subclasses
and documentation of frameworks, 206
and plug-points, 460
vs. subtypes, 115
for types in specs, 245
for versions of behavior of class of objects, 678
Subcomponent partition and specification, 521
Subcomponents
in detailed design, 688
sketching interactions between, 643
Subject areas in system specifications
account, 631
membership, 626–628
and project evolution, 524

rental, 628–629
reservation, 630
Subject class, 676
Subjective models, and meaning of containment, 137–139
Subject-Observer collaboration, 347, 348
Subject-oriented programming, 331
Subroutine, implementation of, 590
Subsets
constraints with, 72
subtypes as, 707
Substitution, 697
explicit, 371
Substitution *(cont.)*
and framework applications, 351, 354
in induction templates, 375
as parameterization, 365
of placeholders, 355, 378
Sorted List framework applied with, 361
Subsystem controller pattern, 494
Subsystems, consistent structure on, 494
Subtype arrows, 370
Subtype constraints, 73
Subtypes, 115
in behavior models, 115
defined, 113, 328
defined as frameworks, 698
defined by templates, 370
and equality, 367
framework applications are not, 345–346
for object models, 560
and refinement, 231
vs. subclasses, 115
as subsets, 707
Subtypes and type extension, 113
attributes and invariants, 113–115
and common pictorial type expressions, 115–117
general type expressions for, 117
no overriding behavior specifications with, 115
Subtype symbol, 231
Subtyping, 322
collaboration refinements beyond, 709
joining type specifications is not, 328–329
for model generalizing, 564
symbol for, 156
Success indicator, 334
Successive refinement, 27
Suffixes, in implementation, 236
Sum abstract class, 244
Sum operators, 65
Superclasses
and documentation of frameworks, 206
implementing methods in, 125
and inheritance, 145
super.method(), 329
Super1::method(), 329
Super2::method(), 329
Superstates, 370
Super/sub interfaces, specifying, 476–477
Supertypes, 231, 323
descriptions by, 564
and equality, 367
loops traversed up to, 569
multiple in behavior models, 116
for object models, 560
and subtypes, 113–115
supply activity, 168, 169
Swimlanes, 700
Swing components (Java), 386
Switch statements, 678
Symbols
to combine sets, 64
for different objects, 53
precise relationships between, 74
for same object, 53
Synchronization parts, and architecture, 491
Synchronous calls, 663
Synchronously invoked actions, 135
Syntax
for applying templates, 364, 365
for Object Constraint Language, 689
Syntropy, 705, 706
joint actions and state chart refinement in, 709
subtyping and viewpoints in, 710
System action specifications
formalizing, 602
state chart information integrated into, 605
System architecture: high-level design, and case study, 529
System behavior specifications, constructing, 596–599
System context, 616
and business model, 518

in case study, 529
defining, 541
use case model of, 588
for video case study, 616
System context diagram, 518
for video case study, 621–626
System context model, *vs*. business model, 624
System contexts, in documentation, 208
System design
architectural elements in, 509
reengineering, 537–538
System detailed design, and case study, 529–530
System models, relationship between design, business and, 625
System operations, user interface scenarios *versus*, 624–626
System requirements/specifications
and business models, 17, 275
reengineering, 537
System specification for video store
account, 631
membership, 626–628
rentals, 628–629
reservation, 630
System specifications
and action reification, 580
and boundary: in typical business system, 519
in case study, 529
implementation of, 532
partitioned into subject areas, 541
patterns for, 530–531
and small groups, 541
specification types in, 525
subject areas in, 626–631
use-case-led, 587–588
for video case study, 615–621, 626–633
System type models, state charts in, 603–606
Szyperski, C., 712

T

Taligent, 711, 713
Targets, supertypes for, 564
Target types, and stereotypes, 371
TCP/IP, 410, 649
Team communication and dynamics, 513
Technical architecture, 209, 409, 516, 524
vs. application architecture, 481, 496–497
and architectural design, 519, 520
design rules for, 524–525
end-to-end implementations of, 513
implementing, 652–653
infrastructure components in, 643
and internal design pattern, 531
and parallel work, 541
and short-cycle development, 540
Technical architecture model, and project evolution, 525
Template collaborations, 29
Template definition, 148
Template method, and framework-style reuse, 462–463
Template packages, 211, 299, 379, 466, 564
and model frameworks, 339, 340, 342
names of, 364
for primitive types, 376–377
to represent inference rules, 374–375
and semantics, 369
theoretical influences on, 711
Template parameters, explicit, 366
Templates, 40
basics with, 373–378
collaboration, 349, 350
for equality and copying, 366–369
as generic types and classes, 364
and higher-level appearance on models, 352
model frameworks as, 13–14
as packages of properties, 359–366
parameterizing, 564
in plug technology, 471
for property connector, 419
provisions in, 360–364
semantic rules expressed as, 372
Temporal logic, 707
Temporaries, and decoupling, 265
tentative Session, 126
Terminal emulators, 393
Terminology
in business models, 558
and model interpretation for clients, 614
Testability
and development-time qualities, 488

and system architecture, 503
Test architecture, 488
Testbeds, 241
and component models, 529
and components, 680
Test harnesses, 233, 250
for component packages, 445
and operation specs, 218, 219
and reusable artifacts, 455
specs turned into, 266, 267
Testing, 237, 272, 709
and abstraction and refinement, 213
Testing *(cont.)*
with abstraction functions, 241–242
action conformance, 249–250
of behavior specifications, 514
in Catalysis, 514
of collaborations as first-class design entities, 708
defined, 215
of development artifacts in package, 710
of invariants, 72
and link and attribute ownership, 664
of nonfunctional requirements, 525
of object and component designs, 214
and operations specifications, 110
and packages, 288
refinement and strong link to, 709
by representing specification in code, 243–246
and retrieve functions, 150
and specifications, 510
technical architecture, 497
theoretical influences on, 709
and well-written postconditions, 86
Test kits, for components, 30
Test sequences, defining, 250
Test specifications, 110
in packages, 316
Text
diagram elements converted to, 370
in narrative of package, 291
Theory extensions, 293
extending type information in, 710
Third-party (or legacy) components, 437, 441, 450–451
Third-party libraries, and architectural views, 493
Third-party packages, 541
Threading services, and EJB tier, 498
Threads
and concurrency, 492–493
defined, 492
and process architecture, 486
Three-valued logic, 706
Threshold, in small components kit, 406
Throughput, 520
TI, and UML, 697
Tidepool, and spiral model, 513
Timeboxing, of common model, 554
Time indexes, 136
Time-line/sequence form, of interaction diagram, 175
Timers, 493
Time-stamping, 317
Time-triggered actions, and context models, 594
Timing constraints, and business rules, 551
To-be business model, implementation of, 510, 532
To-be models, 624
informed by as-is models, 537
making, 536–537
and reengineering, 543
To-be solution models, 523
To-be version business models, 518
Tool dependencies, 653
Tool Requirements, in BluePhone, 303
Tool support, 41
Top-down approach, to refinement, 281
Top-level names, in packages, 288
Tornado, and spiral model, 513
TotalOrdering package, 359, 360, 361
TotalOrdering template, 362
Traceability, 41, 214, 237, 273, 706
and action reification, 580
and business rules, 551
defined, 233
and object technology, 275
and orthogonal abstractions, 278, 279
from problem domain to code, 44
and recursive refinement, 284
and refinement, 279, 510
from spec to code, 640
Trade Supply collaboration, 356
Trade Supply framework, 354, 355
Trade Supply Requirements, 357

Traditional style reuse, *vs.* framework style reuse, 461–465
Training
  for Catalysis adoption, 703
  and project evolution, 526
Traits, 360
Transactional objects, and transaction servers, 525
Transaction Client ports, 416
Transaction commits, 625
Transaction connectors, 425
Transaction objects, 251
Transactions, 412, 413
  with COM+, 403
  and horizontal standards, 396
  specifying, 425
Transaction Server port, 416
Transaction servers, 650
  descriptions of, 525
  and reduction of development work, 542
Transfer connectors, 413, 425
transfer functions, 671
Transfer of workflow objects, 395
Transfer ports, 416
Transfers, 413
  specifying, 425
Transitions
  drawing of, 604–605
  feasible, 131, 132
  pre- and postconditions of labels of, 605
  and reengineering, 536–537
  silent, 133
  simplified by decision points, 129
  between states, 127
  undetermined, 131–132
Transitivity, and package imports, 295
Translation schemes, and package content partitioning, 299
Transparent media, façades as, 647
Tree panels, 500
True | b operator, 68
Turing, ,219
Two-Way Link, Pattern 16.14, 670–671, 676
Two-way links, 259, 673
  and framework building, 352
  and housekeeping message, 658
  and low-level framework, 353
  and role decoupling, 265
Type abstractions, 215
Type-based composition, inheritance replaced with, 475–476
Type boxes, 48
  invariant written within, 70
  meaning of, 324
Type-centric refactoring, 524
Type constants, defining and describing, 65–66
Type-defined equality, 367–368
Type diagrams, 57, 229, 312, 565. *See also* Class diagrams
  action abstraction summarized on, 224
  in business models, 198
  and glossary, 559
  loops in, 568
  for object abstraction, 226
  in packages, 287
  in project glossary, 48
  and reified actions, 251
Typed variables, 145
Type exclusions, in behavior models, 116
Type expressions
  common pictorial, 115–117
  general, 117–118
Type extensions and subtypes, 113–117
Type intersections. *See also* Subtyping
  combining views, 323–324
  defined, 322
Type join, defined, 322
Type libraries, 402
Type model attributes, 99
Type model diagrams, 45
Type Model Is a Glossary, The, Pattern 14.7, 558–559
Type model plus system specs, and project evolution, 523
Type models, 33, 44, 45, 71, 581
  and abstract objects, 256
  for accounting for video store, 631
  and behavior specification, 140
  and business rules, 551
  invariants used in, 72–73
  for membership management, 627
  of Membership Manager component, 682–683
  of Rental Manager component, 680–681
  for rentals from video store, 629
  of Reservation Manager component, 681

for reservations from video store, 630
and snapshots, 613
and system specifications, 519
in video store case study, 571
what they mean and don't mean, 238–239
Type partitioning, in behavior models, 116
Types, 32, 44, 65, 516, 532, 662
action, 8
analyzing, 549
Types *(cont.)*
for architectural models, 484
in architecture, 413
associated with collaborations, 172
attributes of complex object expressed as, 508
in business (domain) models, 528
and business model glossary, 199
vs. class, 82, 114–115
classes implementing, 31
classes representing, 146
and collaboration roles, 25
collection, 689, 691
component architecture, 412–414
decoupling via, 305, 664
defined, 83
defined as frameworks, 698
defined by templates, 370
defined in Catalysis, 695–696
design, 174
in documents, 195
encapsulated collaboration and implementing of, 174
and external behavior of one object, 33–34
in features, 291
implemented by classes, 149
importing of, 305
intersecting, 710
interviews about, 552
links between, 435
marked, 366
model frameworks of, 342–346
nouns sketched as, 552
and object behavior, 82–83, 706
objects defined by, 113
objects described by, 57–58
in packages, 287
parameter, 111
purposes of, 123–124
renaming, 314–315
as roles to decouple, 672, 673
satisfying multiple, 322
as sets, 693
specification, 174
specifications defined by, 196
splitting, 560
and state charts, 603–604
templates as generic, 364
unmarked, 366
uses of, 48, 256
and variables, 145, 265
Type specifications, 139, 154, 581, 703
of calendar, 97–102
combined, for video store, 631–632
defined, 83
joining, 81, 325
joining is not subtyping, 328–329
parts of, 79
summary of, 139–142
and super/sub interfaces, 476–477
for systems, 203
Type theories, 707
Type *vs.* class distinction, and Unified Modeling Language, 697

U

UML. *See* Unified Modeling Language
unassign_instructor, 122
Undefined behaviors, 92
vs. required exceptions, 332
Undefined cases, *vs.* exception cases, 332
Undetermined transitions, 131–132
Unfolding
component diagram based on framework, 419
defined, 344
of framework applications, 14, 343–344, 351
of model frameworks, 341
Ungrouping, 368
Unidirectional dependencies, in packages, 305
Unified Modeling Language (UML), 693, 711
and activity diagrams, 700, 713
attributes, role, and state in, 699

attributes *versus* operations in, 698–699
and capturing rules, 702
class templates in, 341
and collaboration, 701
combining operation specs in, 697
and components, 701
const *versus* {frozen}, 698
dependency between packages in, 296
and enumeration types, 702
and events, 702
and exceptions, 702
extends and used relationships in, 166
and extension, 292
frameworks and, 697–698
and joint actions, 701
and model elements, 297
1.1 standardization of, 697
and packages, 699–700
and parameterized types, 700
and primitives and values, 699
and sets and flat sets, 699
stereotype-based extension in, 711
stereotypes in, 371, 698
and subjective models of containment, 698
syntactical forms in, 370
and type system, 700
type *vs.* class in, 697
unambiguous notation in, 516
and use cases, 700
Uniqueness
constraints, 695
name, 308
Unisys, and UML, 697
Unity, non-OO approaches in, 710
UNIX, 649
pipe-and-filter architecture, 394
UNIX commands, and collaboration, 181
Unmarked types, 366
Unregistering, and framework building, 352
Unsorted queue, 268
Untrue invariants, 106
Up arrow action, 129
update action, 348, 349, 351
Upgradability
with federated scheme, 407
and runtime qualities, 489
Upper interfaces, 458–459
Usability, and runtime qualities, 488–489
Usage imports, circularity and, 296
Use case actions, for video store case study, 572
Use case analysis, 587
Use case diagrams, 165
Use case ellipses, 549, 574
Use-Case-Led System Specification, Pattern 15.3, 581, 587–588, 590, 607
Use case refinement, for video business, 575–579
Use cases, 30, 200, 540, 697, 713
basis for, 165–167
and Catalysis joint action, 701
context model made with, 591–594
defined, 164
exceptions integrated with, 711
extends and uses relationships between, 166
and interaction models, 153
as joint actions, 164–167
refined into finer-grained actions, 543
reifying major concurrent, 644–645
and system context diagrams, 518
in video store case study, 571
Use case specs, 553
Use case templates
documenting refinement textually in, 165
exceptions and, 336–337
Useful constraints, flushing out, 568
User documentation, and project evolution, 526
User interface
and configuration, 499
cross-checking, 542
patterns in, 500
and reuse, 500
and run, 500
and storyboards, 595
User interface bits, and consistency with underlying model, 614
User interface components, 525
User interface domain, 299
User interface modeling, 516
User interface prototyping, 542
User interface reviews, conducting, 614
User interface sketches
and project evolution, 523–524
for video case study, 615–619
User manuals, 510

for existing systems, 549
User roles, and system boundary, 591
Users
bridge requirements for, 585
discussing storyboard with, 595
and interface design, 619
scenarios defined by, 616
User-visible cycles, 540
Uses relationship, and use case, 166
Using State Charts in System Type Models, Pattern 15.9, 531, 582, 603–606
Utility classes
for implementing technical architecure, 653
and infrastructure components, 652
Utting, M., 709

V

VacationScheduler, 25
Validation code, in component packages, 387
Values
for attributes, 50
of navigation expression, 52
and primitives, 699
propagating change in, 253–254
"Vanilla" development, 10
Vanilla process, 713
Variable degree of rigor, 32
Variables
class, 144
features in, 145
instance, 144
local, 144
objects distinguished from, 144
Variants
and factories, 675
vs. versions, 316
Variations, in role decoupling, 673
VBS. *See* Video Business System
VDM
and invariants use, 707
and model and operation refinement, 709
and modeling abstraction of state, 706
model refinement in, 222
Verbs, mapped to actions, 558
Verification and testing
and adapters, 246
and traceability, 233
and well-written postconditions, 86
Verification documents, in packages, 288
Version control, and packages, 315–316, 710
Versions and versioning
and build function, 317
and packages, 285, 318
and separation of business rules, 551
vs. variants, 316
Vertical slices, 281. *See also* Horizontal slices
and package content partitioning, 299, 300, 319
and packages, 299–301
of programs, 188
in short-cycle development, 615
of user-visible functionality, 540
Vertical slices of functionality, visible to end user, 513
Vertical standards, 395, 396
Video business, use case refinement, 575–579
Video Business definition, in video store case study, 572
Video Business System (VBS), specification of, 684, 685
Video-Business_System:: changed action, 628
Video_Business_System:: confirmMember action, 627
Video_Business System:: delete action, 628
Video_Business_System:: purge action, 628
Video Business System Distributed Implementation (VBS-DI), 684, 685, 686
Video case study
abstract business model of, 569–574
component-based design for, 679–688
Videos, distinguished from VideoTitles, 577
Video Store definition, in video store case study, 572
Video Store System (VSS), spec of, 684, 685
VideoTitles, distinguished from Videos, 577
Viewpoint packages, 300
Views
component satisfying multiple, 322, 323
type intersection and combining, 323–324
Virtual abstractions, 229
Virtual imports, 298
Virtual machines
with Java language, 649

and lower interfaces, 460
Virtual packages, 297, 377
and virtual imports, 297
Visibility(ies)
for extensions, 296
and imports, 295
and object refinement, 259
Visual Age (IBM), 408
Visual building tools, 428
and component technology, 408
Visual Café (Symantec), 408
Visual feedback, 524
Visual notations, 428
Vocabulary. *See also* Glossary; Dictionary
and business models, 558
and formalization, 602
value on clear, 708
Vocabulary and rules, representing business, 551

W

Wall-sized documents, 188–189
wanted function, within Reservation Manager component, 681
Web-enabled architecture, four-tiered, 481
Web server tier, in four-tier business architecture, 498
Whiteboarding, and abstract diagrams, 213
White box assets, 455
White-box frameworks, 477
White-box inheritance, 471
“Wide-spectrum” languages, 186
Wills, A. C., 706, 707, 708, 709, 710, 711, 712
Windows, 649
Wirfs-Brock, R., 708, 710
Wong, W., 712
Workflow, 701
system, 563
transfers, 412
Work objects, 409
Workshops, and component specifications, 205
World Wide Web, and hybrid object-relational databases, 502
WRIGHT language, and ABLE project, 712
Writing
abstraction functions, 555
bridge requirements and specifications, 585
business models and requirements documents, 201–202
code, 194–195
component specification documents, 205–206
effect postconditions, 120–121
external action specifications, 179
implementation, 109
internal action specifications, 179
invariants, 67, 70, 108
invariants in collaboration specifications, 180
layered semantics, 377
package expressions, 315
partial specifications, 112
postconditions, 86–87
queries in packages, 316
refinements, 232
retrieval functions, 241
stub operation specifications, 657
type specifications, 599

XYZ

X += y construct, 573
Yourdon, Ed, 187
Z language
and documentation, 708
and invariants use, 707
and modeling abstraction of state, 706
model refinement in, 222, 709
non-OO approaches in, 710
operation refinement in, 709
and schema composition, 707
Zave, Pamela, 709
Zooming in/out, 7, 10
to actions, 223
to software system, 15–16
why abstract and refine?, 214–230