

Index

A

- access control, 184
- account-based security, 327
- ActiveX specification, 120, 377
- actors, 95-98
 - in service interfaces, 71
- aggregation (object relation), 107-109
 - in class diagrams, 56
- Ambler, Scott, 251, 327
- applets, 199
- application (service) interface layer, 16, 29-30
- application databases, 168-170
- application errors, 86-87
- application integration, 158-159
- application mining, 159-160
- applications
 - designing for integration of, 158-159
 - existing, integrating into software design, 51
 - external, represented by objects, 154
 - future of, 382-386
 - in Java, 199
 - services specific to, 78-79
 - workflow tracking applications, 12-13
- applications development, 372-373
- application servers, 6-7, 38
 - alternative architectures for, 34-37
 - architecture of, 16-17
 - for business rule processing, 8-9
 - commercial, 27-28
 - costs and disadvantages of, 9
 - designing, 43-45
 - design of business objects and, 116
 - for distributed processing, 7-8
 - framework for, 178-183
 - implementing security and authorization strategies in, 326-329
 - long-term commitment required for, 10
 - middleware in architecture of, 18-19, 28-29
 - multiprocessing within, 337-340
 - multi-threading and, 346
 - in n-tiered client/servers, 4
 - partitioning of, 149-150
- application service interface objects, 121-122
- architecture
 - of application servers, 16-17
 - of application servers, alternative, 34-37
 - of business objects, 125
 - Distributed interNet Architecture (DNA), 377-378
 - distributed Java versus other middleware, 219-222
 - of Java Virtual Machine (JVM), 198-201
 - of JDBC (Java Database Connectivity), 228-230
 - middleware in, 18-19, 28-29
 - moving from traditional client/servers to n-tiered client/servers, 12-13
 - of n-tiered client/servers, 4
 - for remote procedures, 22-23
- ASCII data files, 168
- association (object relation), 112-113
 - in class diagrams, 55
 - in Java, 203-204
- asynchronous communications, 294-295
- ATMs (automated teller machines), 26
- attributes (properties; instance variables), 101-102, 207
- authorization strategies, 326-329
- automation, 372-373, 376

406 *Building Application Servers*

B

Bassett, Paul, 117–118
 BeanBox (Java framework), 200
 Bea Systems, Inc., 26, 182
 Booch, Grady, 52, 62, 94
 Borland Delphi, 4
 bugs
 debuggers for, 191
 tracking and reporting of, 192
 business object layer, 16, 328
 hidden by services, 81–82
 business objects, 16–17, 30–33
 architecture of, 125
 business rules in, 320
 complex, 255–258, 260–261
 creating new, 122–123
 definition of, 94–95
 design of, 45, 91–92, 101–105
 distributing, 262–263
 handling of, 81–82
 joining with relational databases,
 240–245
 linking to service interfaces, 120–121
 locating, 95–98
 in modeling business processes, 46
 multiprocessing and, 338–339
 multi-threading and, 347
 naming and defining, 98–100
 reuse of, 46–47
 specifications for, 105
 tracking, 245–248
 transition from data models to, 92–93
See also objects
 business processes, modeling, 46
 business rule processing, 6, 8–9
 business rules, 297–299
 classification in, 313–316
 coding, 299–306, 317–321
 commercial business rule engines,
 325–326
 in data, 306–313
 maintaining rule and classification
 tables, 316–317
 security and authorization strategies in,
 326–329

structure-based, 300–301
 byte-code (Java), 199

C

Carey, Bryce, 11
 CASE, *See* computer-aided software engineering
 cell phones, 388–389
 CICS (transaction processing monitor), 26
 class diagrams, 53–58, 115
 classes
 in Java, 201–206
 names for, 206–207
 objects versus, 95
 class factory model, 289, 340–346
 class factory objects, 342–345
 class files, 199
 classifications, 313–316
 classification tables, 314–317
 client/server communications, 270–274
 client/server developers, 4
 client/server software, 16
 Coad, Peter, 94
 COBOL, 157, 164
 Codd, E.F., 132
 coding
 business rules, 299–306, 317–321
 in Java, 206–207
 See also programming
 collections (object relations), 114–115
 COM (Component Object Model;
 Microsoft), 10, 19, 23–25
 comments, in UML diagrams, 59
 commercial application servers, 27–28
 commercial business rule engines,
 325–326
 commercial frameworks for application
 servers, 182
 commercial transaction monitors,
 366–367
 commit method, 363–364
 Common Object Request Broker
 Architecture (CORBA), *See* CORBA
 communications, 375
 asynchronous, 294–295

- client/server, 270–274
 - between computers, middleware for, 18
 - between designers and programmers, 47
 - in development of application server framework, 186–187
 - among objects, 107
 - between persistent objects and databases, 33
 - between user interface and service interface, 29–30
 - compatibility, 153
 - compilers, 190
 - complex business objects, 260–261
 - complex objects, 255–258
 - passing, 288
 - COM+ (Component Object Model; Microsoft), 19, 222, 377
 - component models, 25
 - Component Object Model (COM; Microsoft), 10, 19, 23–25
 - components, of applications, 384–385
 - composition
 - in class diagrams, 56
 - in Java, 203
 - computer-aided software engineering (CASE), 43, 52, 188–189
 - computer crashes, 87
 - computers
 - cheap, 387–388
 - middleware for communications between, 18
 - palm-top computers, 388–389
 - concurrency, 119, 263–266
 - of application server frameworks, 184
 - multiprocessing and, 339
 - concurrency services, 20
 - consultants, 193
 - CORBA (Common Object Request Broker Architecture), 19, 23, 28, 163
 - distributed objects in, 219–221
 - object monitors in, 380
 - objects passed by, 286
 - transaction service in, 366–367
 - costs of application servers, 9
 - C++
 - database wizards in, 137
 - Java based on, 197–198
 - C++ Builder, 4
 - cross-platform integration, 9
 - Crystal Reports, 5
 - culture, organizational, 118–119
 - customer data objects, 82
 - customer objects, 248–250
- D**
- data
 - field validations of, 302–304
 - interdependencies of, 304–306
 - multiple, synchronized, 336
 - multiple sources of, 336
 - in objects, 93
 - primitives, 286
 - relational model of, 132, 133
 - rules in, 306–313
 - service interfaces to locate, 282–284
 - stored in relational databases, 130
 - storing, 284–285
 - synchronizing objects and, 353–360
 - Data Access Objects (DAO; Microsoft), 21, 136
 - databases
 - accessing, 168–170
 - business rule processing for, 6
 - capacity planning for, 259
 - deleting objects from, 264
 - distributed processing of, 7–8
 - JDBC (Java Database Connectivity) for, 200, 233–237
 - locking, 354–355
 - minimizing connections in, 259–262
 - multiple servers for, 262
 - object-oriented databases, 152–153
 - object versus relational, 33
 - optimizing throughput in, 266
 - other middleware for, 238
 - persistence layer for, 32–33
 - persistent objects communicating with, 227–228
 - remote database protocols for, 21
 - Structured Query Language (SQL) for, 230–233
 - See also* relational databases

408 *Building Application Servers*

- database servers, 321
 - deadlocks in, 358–359
 - multiprocessing and, 339–340
 - security implemented in, 328
 - data-centric application servers, 34–35
 - data integrity, 184
 - data models, 92–93
 - data retrieval operations, 76–77
 - Date, C.J., 145
 - deadlocks, in locking, 358–360
 - debuggers, 191
 - delete statement (SQL), 233
 - deleting objects, 140
 - design
 - for application integration, 158–159
 - of application servers, 43–44
 - of business objects, 45, 91–92, 101–105
 - constraints on, 49
 - integrating existing applications into, 51
 - iterative development in, 47
 - joint application design, 44–45
 - layered, 49–50
 - to meet needs of end users, 61–62
 - middleware in, 50
 - modeling business processes in, 46
 - of objects, 93–94
 - of persistent object layer, 137–152
 - programming combined with, 48
 - reuse of objects in, 46–47
 - self-directed technical review in, 48–49
 - of service interfaces, 66–76
 - of services, 76–84
 - standards for, 47
 - Unified Modeling Language (UML)
 - used in, 51–61
 - design patterns, 94
 - destroy method, 285
 - Deugo, Dwight, 389
 - development cycles, 116
 - development environments, 187
 - diagrams (UML), 52–53
 - class diagrams, 53–58, 115
 - sequence diagrams, 58–61, 121–122
 - use case diagrams, 53
 - directory services, 20
 - Distributed Component Object Model (DCOM; Microsoft), 19, 23–24, 28, 163
 - distributed objects in, 221–222
 - objects passed by, 286
 - Distributed Computing Environment (DCE), 19, 21–23, 163
 - Distributed interNet Architecture (DNA; Microsoft), 377–378
 - distributed objects, 23–26, 149–150, 163
 - in CORBA, 219–221
 - debugging, 191
 - in Distributed Component Object Model (DCOM), 221–222
 - distributed processing, 7–8
 - business objects in, 32
 - transaction processing layer in, 26
 - dynamic invocation, 271–272
- E**
- end users
 - design to meet needs of, 61–62
 - multiprocessing by, 332
 - Enterprise JavaBean (EJB) standard, 200–201, 367, 379–380
 - Enterprise Resource Planning (ERP) packages, 383–384
 - error checking, 81
 - error handling, 84–87, 290–294, 322–325
 - in accessing remote objects, 218–219
 - error logs, 324–325
 - error messages, 85, 319
 - errors
 - in database servers, 321
 - fault tolerance and, 185
 - in user interfaces, 318–319
 - See also* exception handling
 - event objects, 104
 - “event occurred” messages, 104
 - events, 104–105
 - Exception class (Java), 290–291
 - exception handling, 84–87, 290–292
 - in accessing remote objects, 218–219
 - by Java, 216
 - in service interfaces, 72–73

- by services, 80–81
 - See also* error handling
- exception objects, 322–323
- execute commands (JDBC), 236
- external applications
 - databases, 168–170
 - integration of, 158–159
 - represented by objects, 154
- F**
- fault tolerance, 185
- field validations, 302–304
- flowcharts, 70
- four-layer architecture, 34
- frameworks, for application servers, 178
 - choosing, 182–183
 - commercial frameworks, 182
 - development strategies for, 185–194
 - initializing, 178–180
 - persistent object frameworks, 238–239
 - requirements for, 183–185
 - service requests processed by, 180–182
- G**
- Gates, Bill, 370
- generalization (object relation), 109–112
 - in class diagrams, 56–57
 - in Java, 206
- generalized object servers, 140–143
- getter and setter methods, 289
- Globally Unified Identifiers (GUIDs), 222
- groupware, 187
- H**
- “handle event” methods, 104–105
- hash tables, 246
- hierarchical classifications, 314–315
- HTML, 37
- I**
- IBM, 27
- IDE (integrated development environment), 190, 199
- impedance mismatches, 144
- infrastructure, 118
- inheritance, 109–112, 141
 - in class diagrams, 56–57
 - in Java, 206
- initializing application server framework, 178–180
- Inprise Application Server, 182
- input and output streams, 164–168
- insert statement (SQL), 231–232
- integration of applications, 158–159
 - application mining for, 159–160
- interdependencies
 - of data, 304–306
 - relational, 310–311
- interface definition languages (IDL), 9, 24–25, 380
 - for communications between user interface and service interface, 29–30
 - Microsoft’s Interface Definition Language (MIDL), 222
 - supported by CORBA, 219–220
- interface objects, 66
- interfaces
 - in Java, to package objects, 207–210
 - remote interfaces, 211–212
 - service interfaces, 65, 66, 269–270
 - services bundled into, 83–84
- Internet, 192, 374–375, 387
 - Distributed interNet Architecture (DNA), and, 377–378
 - Java used on, 197
 - Web server-based tools for, 37
- intranets, 37
- iteration
 - in service interface design, 73
 - in UML diagrams, 59
- iterative development, 47
- J**
- Jacobson, Ivar, 52, 68, 94
- Java (language), 197–198
 - coding guidelines in, 206–207
 - Enterprise JavaBean standard, 200–201, 367, 379–380
 - error handling in, 294

410 *Building Application Servers*

- interfaces to package objects in, 207–210
 - JDBC (Java Database Connectivity), 228–237
 - object-oriented programming in, 201–206
 - other middleware architectures
 - compared with, 219–222
 - remote objects in, 211–219
 - transaction services for, 367
 - JavaBean, 25–26, 120, 200–201, 367, 379–380
 - javac compiler, 199
 - Java Software Developer's Kit (Java SDK)
 - APIs for CORBA in, 29
 - CORBA supported by, 23
 - RMI (Remote Method Invocation)
 - included in, 10, 198
 - RMI registry in, 215
 - Java Virtual Machine (JVM), 198–201
 - threads used by, 334
 - JDBC (Java Database Connectivity), 136, 200
 - architecture of, 228–233
 - programming using, 233–237
 - joint application design (JAD), 43–45
 - common language used in, 73
 - use cases in, 68–69
- K**
- Keuffel, Warren, 11
 - knowledge bases, 325
- L**
- Lammers, Susan, 369–370
 - languages (computer), 190
 - error and exception handling by, 84
 - interface definition languages (IDL), 24–25
 - Java, 197–198
 - Structured Query Language (SQL), 134–136, 230–233
 - supported by CORBA, 219
 - Unified Modeling Language (UML), 51–61
 - languages (human)
- RAD (Rapid Application Development)
 - tools, 190–191
 - used in business objects, 95
 - layered design, 49–50
 - legacy software, 157
 - in development environments, 187
 - proxy objects to represent, 160–162
 - life cycle management, 131
 - life cycle services, 20, 24
 - Linux operating system, 385–386
 - locking, 264, 354
 - at database level, 354–355
 - at object level, 355–357
 - at persistence level, 357–358
 - resolving deadlocks in, 358–360
 - lock method, 361
 - lookup tables, 169–170
 - business rules in, 307–310
 - Lotus Notes, 187
- M**
- managed healthcare applications, 4–6
 - measurements, 194
 - message brokers, 27
 - message handling, 323–324
 - message-oriented middleware (MOM), 165–167
 - messages, 294
 - error messages, 85, 319
 - for errors in user interfaces, 319
 - for events, 104
 - standardized, 322
 - methods, 102
 - names for, 207
 - metrics, 194
 - Microsoft, 15, 370
 - Component Object Model (COM) by, 19, 23–25
 - Component Object Model architecture (COM) by, 10
 - Distributed Component Object Model (DCOM) by, 19, 23–24, 28, 163, 221–222, 286
 - Distributed interNet Architecture (DNA) by, 377–378

- MSMQ message brokers by, 27
- Microsoft Access, 4
- Microsoft Application Services, 378
- Microsoft Foundation Classes (MFC), 141
- Microsoft's Interface Definition Language (MIDL), 222
- Microsoft Transaction Server (MTS), 10, 15, 182, 367
- middleware, 10, 38
 - in application server architecture, 17–19, 28–29
 - categories of, 19–21
 - for databases, 136–137
 - evaluation versions of, 13
 - interface definition languages (IDL), 9
 - message-oriented middleware (MOM), 165–167
 - repositories in, 119–120
 - services provided by, 20
 - in software design, 50
 - vendors of, 15
- modeling
 - business processes, 46
 - Unified Modeling Language (UML) for, 51–61
- MQ-Series message brokers (IBM), 27
- MSMQ message brokers (Microsoft), 27
- multiprocessing, 331–336
 - within application servers, 337–340
- multi-tasking, 333–334
- multi-threading, 150–151, 265, 334, 346–353
- multi-tiered client/servers, *See* n-tiered client/servers
- multi-variant classifications, 315–316
- N**
- naming services, 20, 24, 279
- Netscape, 386
- network errors, 87
- networks
 - accessing remote objects on, 217–219
 - business objects distributed across, 32
 - Distributed interNet Architecture (DNA; Microsoft) and, 377–378
 - distributed processing over, 7–8
 - message brokers for, 27
 - message-oriented middleware (MOM)
 - for, 165–167
 - passing data through, 269–270
 - “sneaker net,” 167–168
 - notation, Unified Modeling Language (UML) for, 51–61
 - notification, 265
 - n-tiered (multi-tiered) client/servers, 4–6
 - designing application servers for, 43
 - moving from traditional client/servers to, 12–13
- O**
- object database management systems (ODBMS), 130, 152–153
 - relational databases versus, 33
- Object Management Group (OMG), 18–19, 386
 - CORBA (Common Object Request Broker Architecture) by, 23
 - transaction specification of, 26
- object modeling, 93
- object monitors, 380
- object-oriented databases, 152–153
- object-oriented programming
 - design combined with, 48
 - in Java, 201–206
- object-oriented programming languages, 32
- object-oriented software design
 - business objects in, 93
 - Unified Modeling Language (UML)
 - standard used in, 51–52
- object relations, 107
 - aggregation, 107–109
 - association, 112–113
 - collections, 114–115
 - generalization and specialization, 109–112
- object request brokers (ORBs), 10
- objects
 - attributes of, 101–102
 - business objects, 16, 30–33, 45, 91–92, 94–95
 - classes versus, 95

412 *Building Application Servers*

- class factory objects, 342–345
 - complex, retrieving, 255–258
 - customer objects, 248–250
 - data in, 93
 - deleting from databases, 264
 - distributed objects, 23–26, 163
 - events and, 104–105
 - external applications represented by, 154
 - increasing, 250–251
 - interactions among, 107–115
 - interface objects, 66
 - in Java, 201–206
 - locking, 355–357
 - methods in, 102
 - multiple, 335–336
 - multiple, retrieving, 251–255
 - packaging, in Java, 207–210
 - passing, 286–288
 - proxy objects, 160–162
 - relational databases and, 144–148
 - remote objects, 211–219
 - reusable business objects, 8
 - reuse of, 117–119
 - states of, 103–104
 - synchronizing data and, 353–360
 - tracking, 143–144
 - transaction objects, 361–363
 - See also* business objects
 - object servers
 - extending, 250–258
 - generalized object servers, 140–143
 - persistent object servers, 239–250
 - ODBC (open database connectivity; Microsoft), 21, 136, 238
 - JDBC (Java Database Connectivity)
 - bridge to, 200, 229–230, 234–235
 - Open JORBUX, 386
 - Open Software Foundation (OSF), 18–19
 - Distributed Computing Environment (DCE) by, 21–23
 - open source software, 385–386
 - operating systems
 - Linux, 385–386
 - Multi-tasking by, 333
 - multi-threading and, 346–347
 - order lookup screens, 79
 - organization-based security, 327
- P**
- Page, John, 370
 - palm-top computers, 388–389
 - parameters, 272
 - objects passed as, 286
 - in UML diagrams, 59
 - parent objects, 109
 - persistence, 120, 144, 321
 - locking, 357–358
 - persistence layer, 16–17, 32–33, 250, 321
 - multiprocessing and, 339
 - optimizing, 258–259
 - role of, 130–131
 - persistence services, 20
 - persistent object frameworks, 238–239
 - persistent object layer, 129–130, 227, 239
 - design of, 137–140
 - generalized object servers in, 140–143
 - relational databases and, 144–148
 - scalability of, 149–152
 - tracking objects in, 143–144
 - persistent objects, 130
 - communicating with databases, 227–228
 - loading into memory, 179
 - multiprocessing and, 339
 - persistent object servers, 239–250, 260
 - pervasive computing, 389
 - Petzold, Charles, 207
 - platforms, 374, 387
 - cross-platform integration, 9
 - Java Virtual Machine (JVM) as, 199
 - primitives, 286
 - procedures
 - remote procedure calls for, 21–23
 - in service interfaces, 71–72
 - processes, 118
 - multiprocessing of, 332
 - programming
 - design combined with, 48
 - multi-tiered client/servers, 11
 - tools for, 188–192

- using JDBC (Java Database Connectivity), 233–237
 - programming languages and tools, 190
 - project management software, 188
 - properties, 288–290
 - property pages, 289–290
 - proxy objects, 160–162
 - punch cards, 171
- R**
- RAD (Rapid Application Development)
 - tools, 35–36, 190–191, 381
 - Rational Rose (CASE tool), 58
 - Raymond, Eric, 385
 - reference counters, 246–247
 - relational databases, 130, 132
 - joining business objects with, 240–245
 - middleware for, 136–137
 - object databases versus, 33
 - objects and, 144–148
 - relational data model for, 133
 - Structured Query Language (SQL) for, 134–136, 230–233
 - See also* databases
 - relational data model, 132, 133
 - remote communications, 271–272
 - remote database protocols, 21
 - remote interfaces, 211–212
 - remote job entry (RJE) software, 171
 - remote objects, 211–219
 - releasing, 285–286
 - remote procedure calls, 21–23, 162–163
 - remote software, 162–164
 - replication of database changes, 169–170
 - repositories, 119–120
 - resource locking, 354
 - reusable business objects, 8, 91
 - reusable software, 12
 - reuse of objects, 46–47, 117–119
 - RMI (Remote Method Invocation), 10, 23, 28–29, 198
 - to distribute Java objects, 211–219
 - JDBC (Java Database Connectivity) and, 200
 - RMI naming service, 341
 - role-based security, 327
 - rollback method, 365, 366
 - rule objects, 320
 - rule processing, 297
 - rules, *See* business rules
 - rule tables
 - explicit, 311–312
 - generalized, 312–313
 - Rumbaugh, James, 52
- S**
- scalability, 149–153
 - of application server frameworks, 183–184
 - screens
 - order lookup screens, 79
 - screen scraping, 164
 - security, 326–329
 - in application server frameworks, 184–185
 - security objects, 328
 - security services, 20
 - select statement (SQL), 231
 - self-directed technical review, 48–49
 - sequence diagrams, 58–61, 121–122
 - sequential pooled connections, 261–262
 - servers
 - communications by, 272–273
 - database servers, 321
 - multiple, 262
 - in n-tiered client/servers, 4
 - service (application) interface layer, 16, 29–30
 - service interface objects, 121–122
 - service interfaces, 65, 66, 269–270
 - actors in, 71
 - business rules in, 319–320
 - context of, 69–70
 - creating, 274–280
 - defining, 275–276
 - design of, 66–68
 - error and exception handling by, 84–87
 - exceptions in, 72–73
 - implementing, 276–279
 - linking to business objects, 120–121

414 *Building Application Servers*

- multiprocessing and, 338
 - procedures in, 71–72
 - registering, 279–280
 - security implemented in, 328
 - use of, 280–285
- services
 - accessing, 281–282
 - application-specific, 78–79
 - bundling into interfaces, 83–84
 - business object layer hidden by, 81–82
 - conforming to standards, 82–83
 - exceptions handled by, 80–81
 - implementing, 123–125
 - processing requests for, 180–182
 - requesting, 274
 - self-contained, 79–80
 - subroutines turned into, 160
 - use cases turned into, 76–78
- Simonyi, Charles, 207
- skeleton programs, 24, 29–30, 214, 272–273
- “sneaker net,” 167–168
- socket APIs, 164
- software
 - legacy software, 157
 - reusable, 12
- software design, *See* design
- software designers, 7
- source code, generated by CASE, 189
- specialization (object relation), 109–112
- standards, 373, 376–377
 - CORBA (Common Object Request Broker Architecture) as, 23
 - in design, 47
 - Distributed Computing Environment (DCE) as, 21–23
 - Distributed interNet Architecture (DNA), 377–378
 - Enterprise JavaBean, 200–201, 379–380
 - for middleware, 18–19
 - for notation, Unified Modeling Language (UML) as, 52
 - in programming, 11
 - services conforming to, 82–83
 - states, 103–104
 - static invocation, 271, 272
 - store method, 361
 - Structured Query Language (SQL), 134–136, 230–233
 - used in JDBC (Java Database Connectivity), 236–237
 - stub programs, 24, 29–30, 214, 271, 272
 - subroutines, 160
 - Sun Microsystems
 - CORBA supported by, 23
 - Enterprise JavaBean standard by, 200–201, 367, 379–400
 - Java language by, 197
 - JDBC-ODBC bridge driver by, 229–230
 - symbols (UML), 52
 - See also* diagrams (UML)
 - synchronization, 119, 263–266
 - of objects and data, 353–360
 - synchronizing transactions, 170
 - system administrators, 7
 - system errors, 87
- T**
- table locking, 354
- tables
 - classification tables, 314–317
 - in databases, 133
 - hash tables, 246
 - lookup tables, 169–170, 307–310
 - rule tables, 311–313
- testing tools, 191
- Thorpe, Margaret, 298
- threads, 334
- three-tiered client/servers, 4
- throw command, 290, 292
- time services, 20
- tools, 188–192
- Torvalds, Linus, 385
- tracking objects, 143–144
- training, 192–194
- transaction objects, 361–363
- transaction processing layer, 26, 34
- transaction processing monitors, 26–27, 33
 - commercial transaction monitors, 366–367
- transaction requests, 76

-
- transactions, 151–152, 360–367
 - synchronizing, 170
 - trial projects, 12–13
 - try/catch blocks, 290, 292–293
 - Tuxedo (transaction processing monitor), 26
 - two-phase commit process, 365–366
 - two-tiered client/servers, 3–4
 - moving to n-tiered client/servers from, 12–13
- U**
- Unified Modeling Language (UML), 43, 51–52
 - diagrams and symbols in, 52–61
 - unlock method, 361–362, 364
 - update statement (SQL), 232–233
 - use case diagrams, 53
 - use cases, 68–69
 - in service interface design, 76–78
 - user interfaces, 16
 - application (service) interface layer and, 29–30
 - business rules in, 318–319
 - designing, 76–77
 - error handling by, 85–86
 - multiprocessing and, 338
 - security implemented in, 328–329
 - users, *See* end users
- V**
- variables, multi-threading and, 334
 - version control, 189–190
 - Visual Basic, 384
- W**
- waterfall method of software design, 44
 - Web browsers, 199–200
 - Web pages, 199
 - Web server-based tools, 37
 - Windows
 - Distributed Component Object Model (DCOM) on, 23–25, 222
 - names for classes in, 206–207
 - ODBC (open database connectivity)
 - used with, 238
 - Windows 2000, 19
 - word processing, 372
 - workflow tracking applications, 12–13
- Y**
- Y2K problem, 301
 - Yourdon, Ed, 94