

Index

- a-periodic messages, 98
- a-periodic tasks, 121
- Aalto, Alvar, 536
- Abnormal network conditions, 167
- ABSOLUTE OBJECT REFERENCE pattern, 393
- AC (Authentication Center), 226
- Access agents, 340, 343–345
- Account class, 4–6, 17–19
- AccountType class, 4–6, 17, 19–20
- Acrobat plug-ins, 304
- Action class, 55
- Activation time of plug-ins, 323
- Active Directory, 200
- Active Object Model. *See* DYNAMIC OBJECT MODEL pattern
- Active Registration, 320
- Actiweb system, 291–292
- ADAPTERS pattern
 - COMPONENT WRAPPER, 370
 - VOID, 502
- ADAPTIVE OBJECT MODEL pattern, 375
- Adjectives, 444
- ADP templates, 278
- Advanced pattern writing
 - acknowledgments, 451
 - CONSISTENT “WHO”, 446–448
 - DEAD WEASELS, 443–444
 - existing work, 434–436
 - FORCES HINT AT SOLUTION, 444–446
 - “HOW”-PROCESS, 438–440
 - introduction, 433–434
 - POINTERS TO DETAIL, 448–451
 - references, 451–452
 - “WHY”-PROBLEMS, 440–443
- Adverbs, 444
- Agent module, 350
- AGGREGATION pattern, 461, 494–497
 - DYNAMIC OBJECT MODEL, 12–13
 - RESOLUTION OF FORCES, 483
- Aha effect, 519–521
- Air interface, 217
- Airplane Information Management System (AIMS), 122
- Alarm class, 318
- Alexander, Christopher
 - biological analogy by, 464
 - on city structure, 488
 - on complex systems, 455–456, 489
 - on context, 498
 - on designed systems, 454–455
 - on differentiation, 473, 492
 - on emptiness, 500
 - on entrances, 481

- Alexander, Christopher, *continued*
 on evolution, 501
 on forces, 461–462, 468–469, 472, 480–483, 488, 498
 on fundamental process, 463
 on HALF-HIDDEN GARDEN, 484
 and LDPL, 453, 458
 motivation of, 552
 on paths, 499
 on pattern growth, 495–496
Pattern Language, A—Towns, Buildings, Construction, 259
 on pattern sequences, 464–465
 on scale, 485–486
- Alexandrian pattern form, 437–438
 Prairie Houses, 534–537
 Web content conversion and generation, 259
- ALIGN DIRECTORIES WITH ORGANIZATION pattern, 199–200
 CONSULT DIRECTORY, 196
 OVERLAY NETWORK, 193
 SERVER DOES HEAVY WORK, 205
- Ambiguous comparands, 180
- Analog mobile wireless systems, 227
- Analysis Patterns* (Fowler), 22
- Analyzers, 309
- Anchor entities
 in mobile wireless systems, 220, 242–244
 summary, 255
- “And They Lived Happily Ever After” syndrome, 523
- Andrade, Rossana Maria de Castro
 biographical information, 555
 MoRaR, 213
- Anti-patterns, 520
 Blob, 53, 58–59
 Poltergeist, 47
- AOL Server, 293
- Apache Axis framework, 381, 396–397
- Appleton, Jay
 on prospect and refuge, 533
 on symmetries, 484–485
- Application/Communication Buffer pattern. *See* TEMPORAL APPLICATION DECOUPLING pattern
- Application servers, 32
- Application Submission and Control Tool (ASCT) module, 351
- Application tasks, 116–117
- Applications
 configurable. *See* Plug-ins
 deployment of, 351–352
 development of, 102
- Apptimizer system, 60
- Architectural concepts for MoRaR, 215–218
- Architectural pattern languages, 299–300, 458
 CLUSTER OF FORCES, 469
 COMMON GROUND, 498
 CROSS LINKAGES, 489
 HALF-SYNC/HALF-ASYNC, 120
 LEVELS OF SCALE, 486–487
 RESOLUTION OF FORCES, 481–482
 VOID, 501
- Arguments Object pattern, 62
- ARINC 659 standard
 BIUs in, 115
 in hard real-time systems, 122
 in SYNC FRAME, 111
 in TEMPORAL APPLICATION DECOUPLING, 119
- Artix framework, 381
- ASCT (Application Submission and Control Tool) module, 351
- ASK LOCAL NETWORK pattern, 191–192
 CONSULT DIRECTORY, 196
 LISTEN TO ADVERTISEMENTS, 194

- PLACE DIRECTORIES DYNAMICALLY, 201
- SEPARATE IDENTITY FROM LOCATION, 202
- SERVICES REGISTER IN DIRECTORY, 197
 - in SLP, 206
 - USE ADVERTISER, 195
- ASP pages, 293
- Aspect configuration, 390–393
- Aspect ratio in Prairie Houses, 538
- Asynchronous-Change Commands
 - pattern, 84–85
- Attribute reuse, 178
- Audiences, 447
- Authentication
 - in mobile wireless systems, 215, 220, 225–227, 229–232
 - summary, 254
- Authentication Centers (ACs), 215, 226
- Authenticator pattern, 232
- Author as Owner pattern, 508, 517–518
- AUTOMATIC CONTROLS pattern
 - in Equitable Resource Allocation, 146
 - in Real Time and Resource Overload language, 132, 148
- Automatic Invocation application, 323
- AUTOMATIC OUT-OF-CHAIN ROUTING
 - pattern
 - in Equitable Resource Allocation, 147
 - in Real Time and Resource Overload language, 132, 148
- AUTOMATIC TYPE CONVERTER pattern, 359–360, 377–381
 - COMMAND LANGUAGE, 365
 - COMPONENT WRAPPER, 370
 - CONTENT CONVERTER, 273
 - SPLIT OBJECT, 383, 386, 390, 397
- Automation, DYNAMIC OBJECT MODEL
 - for, 10
- Availability in distributed systems, 157
- Available bandwidth for a-periodic transmissions, 98
- Axis framework, 381, 396–397
- BACKEND pattern, 30, 35
- BALANCED CONTEXT pattern, 509, 523–525
- BANDS OF WINDOWS pattern, 539, 545–547
- Bandwidth utilization
 - in hard real-time systems, 96
 - in PRESCHEDULED PERIODIC TRANSMISSION, 98, 101
 - in SYNC FRAME, 110
- Banking systems. *See* DYNAMIC OBJECT MODEL pattern
- Bar-Yam, Y., 483
- Base station controllers (BSCs), 216–218
- Base station transceivers (BSTs), 217–218
- Base stations, 217
- Bathing Room pattern, 500
- BEA WebLogic Integration, 291
- BEAUTY OF SIMPLICITY pattern, 407–411, 413, 418
- Behavioral patterns in MoRaR, 218–221
- BIG PICTURE pattern, 436, 438, 508–509, 515–516
- Biological systems
 - AGGREGATION, 497
 - CROSS LINKAGES, 491
 - DIFFERENTIATION, 493–494
 - in LDPL, 458, 464, 467
 - LOCAL SYMMETRIES, 477–478
 - VOID, 502

- Bit masks, 141–142
- BIUs (Bus Interface Units), 115, 119
- Blob antipattern, 53, 58–59
- Bottlenecks, 58
- Boundary conditions, 177
- Brake control systems. *See* Hard real-time systems
- Brand, Stewart, 463
- BROKER pattern, 345, 353
- Browsers
 - FRAMEWORK PROVIDING APPLICATION, 318
 - PLUG-IN, 309
- BSCs (base station controllers), 216–218
- BSP API, 348
- BSTs (base station transceivers), 217–218
- BUILD TRUST pattern, 423
- Bus cycles, 102
- BUS GUARDIAN pattern
 - consequences, 114
 - context, 111
 - examples, 112, 114
 - in hard real-time systems, 92, 97
 - implementation, 113–114
 - known uses, 115
 - problem, 111
 - related patterns, 115
 - solution, 112
 - in SYNC FRAME, 109
- Bus Interface Units (BIUs), 115, 119
- Business logic, 25–26, 32
- Business rules engine, 21
- Buy class, 47
- Byzantine situations, 109, 111
- C language
 - AUTOMATIC TYPE CONVERTER, 381
 - dispatching in, 79
 - OBJECT SYSTEM LAYER, 376
- C++ language
 - dispatching in, 79
 - encapsulation in, 50, 60–61
 - OBJECT SYSTEM LAYER, 374
 - and OTcl, 389–390
 - sameness in, 177, 183
 - wrappers in, 371–373
- C++ Idioms Language, 459
 - AGGREGATION, 496
 - COMMON GROUND, 499
 - CROSS LINKAGES, 489–490
 - DIFFERENTIATION, 493
 - LOCAL SYMMETRIES, 474
 - operator overloading in, 465
- Calder, Paul
 - biographical information, 555
 - LDPL, 453
- Camargo, Raphael Y. de
 - biographical information, 555–556
 - GRID, 337
- CANTILEVERED TERRACE pattern, 543–545
 - BANDS OF WINDOWS, 546
 - FIREPLACE AS REFUGE, 547
 - for horizontal spread, 539
- Car brake control systems. *See* Hard real-time systems
- Carneiro, Marcio
 - biographical information, 556
 - GRID, 337
- CAVE (Cellular Authentication and Voice Encryption), 232
- Cells in mobile wireless systems, 215
- Central Bus Guardian, 113–114
- CHAIN OF RESPONSIBILITY pattern
 - CONTENT CONVERTER, 271–272
 - DYNAMIC OBJECT MODEL, 13
 - ENCAPSULATED CONTEXT, 61
- Change requests in framework development, 423–425

- Channel-specific formats, 264
- CheckingAccount class, 17
- Checkpointing, 349
- CHECKS pattern language, 459, 466–467
 - CLUSTER OF FORCES, 470
 - CROSS LINKAGES, 490–491
 - LEVELS OF SCALE, 487
 - RESOLUTION OF FORCES, 482
 - VOID, 501
- CHI (controller host interfaces), 119
- CHIMNEY AS ANCHOR pattern, 539, 541–543
 - CANTILEVERED TERRACE, 545
 - FIREPLACE AS REFUGE, 547
- Ciphering
 - in mobile wireless systems, 215, 220, 223, 226–229
 - summary, 253
- CIRCUITOUS APPROACH pattern, 539, 541, 550–551
- City is Not a Tree, A* (Alexander), 488
- Classes, comparable, 176–177
- CLEAR TARGET AUDIENCE pattern, 435, 447, 525
- CLEAVAGE pattern, 477–478
- Client class, 9
- CLIENT-DISPATCHER-SERVER pattern, 203
- CLIENT KNOWS BEST pattern, 203–204
- Clients
 - DECENTRALIZED LOCKING, 157–158
 - in Service Discovery, 189
- Cliques in SYNC FRAME, 109
- Clock synchronization, 103–105
- Clone comparisons, 175–176
- CLUSTER OF FORCES pattern, 460, 467–471
- CNI (communication network interface), 119
- Cocoon framework, 426
- CODE SAMPLES pattern, 436, 449
- CODE SAMPLES AS BONUS pattern, 436, 449
- Collaborations, 7–8, 30–31
- Collection Framework, 177
- Collections
 - COMPARAND, 183–184
 - GRID, 350
- Collisions on Ethernet, 94–95
- COM (Component Object Model), 182
- COMMAND pattern, 359–360
 - ENCAPSULATED CONTEXT, 47, 61
 - substitutions in, 363
- COMMAND LANGUAGE pattern, 358–360
 - AUTOMATIC TYPE CONVERTER, 377, 379–380
 - COMPONENT WRAPPER, 370
 - consequences, 366–367
 - context, 361
 - discussion, 363–366
 - forces, 362
 - known uses, 367
 - OBJECT SYSTEM LAYER, 374
 - problem, 361–362
 - scenarios, 362, 366
 - solution, 363
 - SPLIT OBJECT, 382, 384–385, 387–389, 391, 393–394
- COMMANDS pattern, 358–360
 - COMMAND LANGUAGE, 365
 - COMPONENT WRAPPER, 370
- Comments in shepherding, 514
- COMMON AREA AT THE HEART pattern, 499
- COMMON GROUND pattern, 497–500
 - in LDPL, 461
 - LEVELS OF SCALE, 487
 - RESOLUTION OF FORCES, 483
- Communication
 - in DISASTER NOTIFICATION, 135
 - in GRID, 343, 348

- Communication handlers, 116–117
- Communication network interface (CNI), 119
- COMMUNITY OF TRUST pattern, 513
- COMPARAND pattern
 - acknowledgments, 187
 - conclusion, 186–187
 - consequences, 183–184
 - context, 170
 - example, 169–170
 - implementation, 174–183
 - known uses, 184–186
 - overview, 169
 - problem, 171–173
 - references, 187–188
 - related patterns, 186
 - solution, 173–174
- ComparandFactory class, 176
- Comparisons
 - operations, 178–179
 - semantics, 174
- Complex systems, pattern languages
 - as, 455–456
- Complexity
 - in BUS GUARDIAN, 114
 - COMPARAND, 183
 - DYNAMIC OBJECT MODEL, 6–7
 - in framework development, 408, 416
 - GRID, 352
 - in SYNC FRAME, 110
- Component and language integration, 357
 - acknowledgments, 398
 - AUTOMATIC TYPE CONVERTER, 377–381
 - COMMAND LANGUAGE, 361–367
 - COMPONENT WRAPPER, 368–372
 - conclusion, 397
 - introduction, 358–360
 - OBJECT SYSTEM LAYER, 372–376
 - patterns, 360–361
 - references, 398–400
 - SPLIT OBJECT. *See* SPLIT OBJECT pattern
- Component class, 7, 9
- Component Object Model (COM), 182
- COMPONENT WRAPPER pattern, 368–372
 - implementation, 359
 - OBJECT SYSTEM LAYER, 373
 - SPLIT OBJECT, 394, 397
- COMPONENT WRAPPERS pattern
 - AUTOMATIC TYPE CONVERTER, 359
 - OBJECT SYSTEM LAYER, 374
 - SPLIT OBJECT, 382, 386, 388
- Components
 - in hard real-time systems, 120
 - vs. plug-ins, 302–303
- ComponentType class, 7, 9, 14
- Composability, 101
- COMPOSITE pattern
 - DYNAMIC OBJECT MODEL, 22
 - FRAGMENTS, 281, 284
 - GENERIC CONTENT FORMAT, 265, 267
- Compound comparands, 180–182
- CompoundContent class, 266
- Computations, grid, 338–339
- Computed comparands, 182
- CONCEALED VERTICALS pattern, 549–550
- CONCRETE EVIDENCE FOR REUSE pattern, 404–407, 425
- CONCRETEFINDER pattern, 30
- CONCRETEMANAGER pattern, 30, 34
- Concurrency in reference counting, 79
- Condor Grids, 350
- Configurable software. *See* Plug-ins
- “Connecting Business Object to Relational Databases” (Yoder), 40

- Connection interface, 30
- CONSISTENT “WHO” pattern, 434, 446–448
- Constraints
 - DYNAMIC OBJECT MODEL, 13
 - GRID, 346
 - in PRESCHEDULED PERIODIC TRANSMISSION, 97
 - in TEMPORAL APPLICATION DECOUPLING, 119
- CONSULT DIRECTORY pattern, 195–196
 - CONSULT DIRECTORY, 196
 - LISTEN TO ADVERTISEMENTS, 194
 - SEPARATE IDENTITY FROM LOCATION, 202
 - in SLP, 206
 - USE ADVERTISER, 195
- Consumer/user behavior in OVERLOAD EMPIRES, 134
- CONTENT CACHE pattern, 257, 264
 - CONTENT CONVERTER, 272
 - FRAGMENTS, 283
 - overview, 262
 - PUBLISHER AND GATHERER, 268
 - Web content conversion and generation, 284–287
- CONTENT CONVERTER pattern, 257
 - AUTOMATIC TYPE CONVERTER, 380
 - CONTENT CREATOR, 275
 - GENERIC CONTENT FORMAT, 266
 - overview, 261
 - Web content conversion and generation, 270–274, 289
- CONTENT CREATOR pattern, 257, 260
 - CONTENT CONVERTER, 273
 - CONTENT FORMAT TEMPLATE, 279
 - overview, 261
 - Web content conversion and generation, 274–277, 290
- CONTENT FORMAT TEMPLATE pattern, 257, 260
 - overview, 262
 - Web content conversion and generation, 277–279
- Context, encapsulated. *See* ENCAPSULATED CONTEXT pattern
- Context Beans, 61
- Context class, 51–59
- Controller host interfaces (CHI), 119
- Conversions
 - AUTOMATIC TYPE CONVERTER. *See* AUTOMATIC TYPE CONVERTER pattern
 - content. *See* CONTENT CONVERTER pattern
 - Web. *See* Web content conversion and generation
- CONVINCING SOLUTION pattern, 509, 519–521
- COOPERATING PLUG-INS pattern, 306, 310, 327–330
- Coordinated comparands, 182–183
- Copies
 - COMPARAND, 172
 - ENCAPSULATED CONTEXT, 50–51, 57, 61
- Coplien, Jim
 - C++ Idioms Language, 459, 465
 - ENGAGE CUSTOMERS, 415
 - “Pattern Language for Writers’ Workshops”, 508
 - on pattern languages, 455
 - SIZE THE ORGANIZATION, 413
 - on symmetry breaking, 457
 - THREE ITERATIONS, 512
- COPY-ON-DEMAND pattern, 82–84
- COPY-THEN-DISPATCH pattern, 81–82
- CORBA Object Adapter component, 71–73

- CORBA Object Factories component, 39
- CORBA Relationship Service, 185
- CorbaSEC security, 348
- CosObjectIdentity module, 185
- Costanza, Pascal
 - biographical information, 556
 - COMPARAND, 169
- Costs in BUS GUARDIAN, 114
- COUNTED BODY pattern, 466
 - AGGREGATION, 496–497
 - COMMON GROUND, 499
 - DIFFERENTIATION, 493
 - LOCAL SYMMETRIES, 475–476
- Counting references in dispatch, 78–80
- Coupling encapsulated context, 50, 57
- COURTYARDS WHICH LIVE pattern, 489
- CPU utilization, 119, 133
- Credibility, 520
- Credit Control Platform, 292
- CROSS LINKAGES pattern, 460, 487–491
- CRUD pattern, 40
- Cunningham, W., 482
- Czarnecki, K., 473
- D-AMPS (Digital Advanced Mobile Phone System), 213
- DA (Directory Agent), 196
- Dasgupta, S., 456
- Data access layer framework
 - background, 402–403
 - evidence for reuse, 406
 - framework user involvement, 422
 - multiple change requests, 424–425
 - pilot applications, 415
 - pilot-based tests, 419–420
 - simplicity, 409–410
 - skilled teams, 412
 - small objects, 417–418
- Data conversion
 - AUTOMATIC TYPE CONVERTER. *See* AUTOMATIC TYPE CONVERTER pattern
 - content. *See* CONTENT CONVERTER pattern
 - Web. *See* Web content conversion and generation
- Data copying, 50–51, 57, 61
- Data Router system, 60
- Data Update Propagation (DUP) algorithms, 282
- Data Visualizers, 292
- Databases
 - in mobile wireless systems, 215, 220, 223, 225–226, 235–237
 - in paging, 233–234
 - summary, 253–254
- DEAD WEASELS pattern, 434, 443–444
- Debugging
 - COMPARAND for, 170
 - GRID for, 352
- DECENTRALIZED LOCKING pattern
 - acknowledgments, 167–168
 - consequences, 166–167
 - context, 156
 - dynamics, 159–161
 - example, 155–156, 164
 - implementation, 161–164
 - introduction, 155
 - known uses, 165–166
 - problem, 156–157
 - references, 168
 - related patterns, 167
 - solution, 157
 - structure, 157–158
 - variants, 165
- Decentralized start-up, 110
- DECORATOR pattern
 - COMPARAND, 171, 175
 - COMPONENT WRAPPER, 370

- Deferred plug-in loading, 308
- DeLano, David, 515
- DEPARTMENT HEARTH pattern, 498
- Dependency Injection technique, 320
- Dependent Demand pattern, 22
- Deployment structure in DECENTRALIZED LOCKING, 157
- Design Patterns: Elements of Object-Oriented Software Design*, 526
- Designed systems, pattern languages as, 454–455
- Destruction in encapsulation, 50
- DETACHED COUNTED BODY pattern, 466
 - AGGREGATION, 496–497
 - COMMON GROUND, 499
- Device drivers, 309
- DIFFERENTIATION pattern, 491–494
 - LEVELS OF SCALE, 487
 - purpose, 461
 - RESOLUTION OF FORCES, 483
- Digital Advanced Mobile Phone System (D-AMPS), 213
- Directory Agent (DA), 196
- DIRECTORY FINDS SERVICES pattern, 197–199
- DISASTER NOTIFICATION pattern
 - in Equitable Resource Allocation, 147
 - in Real Time and Resource Overload language, 134–136
- Discovering services. *See* Service discovery
- Dispatching components, 69
 - conclusion, 85–86
 - introduction, 69–70
 - multiple objects, 80–85
 - overview, 70–73
 - references, 86–88
 - single objects, 74–80
 - SPLIT OBJECT, 383
- Distributed environments, 179–180
- Distributed Locking. *See* DECENTRALIZED LOCKING pattern
- Distributed object computing (DOC)
 - middleware, 69–70
- Distributed processing capabilities.
See GRID pattern
- Distributed resources, 352
- Distributed systems, 153–154, 173
- Disturbances in PRESCHEDULED PERIODIC TRANSMISSION, 102
- DNS
 - ALIGN DIRECTORIES WITH ORGANIZATION, 200
 - SEPARATE IDENTITY FROM LOCATION, 202
 - SERVICES REGISTER IN DIRECTORY, 198
- Doble, Jim, “Pattern Language for Pattern Writing”, 434
- DOC (distributed object computing)
 - middleware, 69–70
- DocMe system, 292
- Document archiving systems
 - SPLIT OBJECT, 386
 - Web content conversion and generation, 292
- DOM (Document Object Model), 289, 291
- Domain names, 202
- DOMAIN OBJECT MANAGER pattern, 25
 - acknowledgments, 41
 - applicability, 28–29
 - collaborations, 30–31
 - consequences, 32
 - implementation, 32–35
 - intent, 25
 - known uses, 39
 - motivation, 25–28
 - participants, 29–30
 - references, 42–43

- DOMAIN OBJECT MANAGER pattern,
 - continued*
 - related patterns, 40
 - sample code, 35–38
 - structure, 29
- Domain-specific languages, 7, 367
- Domain-specific patterns, 211–212
- DOMAINOBJECT pattern, 30, 32–33
- Dominant siblings, 59–60
- DUP (Data Update Propagation)
 - algorithms, 282
- Dynamic behavior, 11–12
- Dynamic consumer subscriptions, 73
- Dynamic languages, wrappers for, 383
- Dynamic loading of plug-ins, 323
- DYNAMIC OBJECT MODEL pattern, 3
 - acknowledgments, 22
 - dynamic behavior, 11–12
 - end-user configuration, 11
 - extensions, 12–13
 - flexibility, 10
 - implementation, 13–15
 - known uses, 20–21
 - motivation, 3–6
 - portability, 12
 - problem, 6–7
 - programming environment, 11
 - references, 22–24
 - related patterns, 22
 - runtime typing, 12
 - sample code, 15–20
 - simplicity, 9–10
 - solution structure, 7–9
- DYNAMIC OVERLOAD CONTROL pattern
 - DISASTER NOTIFICATION, 136
 - in Real Time and Resource Overload language, 132, 148
- ECHO BACK pattern, 487
- Eckstein, Jutta, 515
- Eclipse application plug-ins, 304, 309
 - PLUG-IN BASED PRODUCT, 332
 - PLUG-IN CONTRACT, 315
 - PLUG-IN LIFE CYCLE, 324
 - PLUG-IN REGISTRATION, 321
- Edge Side Includes, 293
- Editing databases, 21
- Eisenecker, U. W., 473
- Electro Mechanical Brake (EMB) systems. *See* Hard real-time systems
- Elevation in Prairie Houses, 541
- Embedded systems, 90
- Emergency braking, 92
- Emptiness, 500
- ENABLING THE SYSTEM TO SHARE LOAD pattern, 133
- ENCAPSULATED CONTEXT pattern
 - acknowledgments, 63–64
 - audience, 45
 - consequences, 56–60
 - examples, 45–48, 63
 - forces, 49–51
 - implementation, 52–54
 - known uses, 60–61
 - problem, 49
 - references, 64–65
 - related patterns, 61–63
 - resolution, 54–55
 - solution, 51–52
 - summary, 63
 - variations, 55–56
- Encapsulation of heterogeneity, 351
- Encryption, 223, 226–229
- End-user configuration in DYNAMIC OBJECT MODEL, 11
- ENGAGE CUSTOMERS pattern, 415
- Enterprise Java Beans
 - COMPARAND, 185
 - DOMAIN OBJECT MANAGER, 39
 - ENCAPSULATED CONTEXT, 61

- Entities in mobile wireless systems, 220
- ENTRANCE TRANSITION pattern
 - CIRCUITOUS APPROACH, 551
 - CONCEALED VERTICALS, 550
 - forces for, 469–470, 481–482
- Equation constraints, 119
- Equitable manner, 132
- Equitable Resource Allocation
 - OVERLOAD EMPIRES, 133
 - in Real Time and Resource Overload language, 146–147
- Eronen, Pasi
 - biographical information, 556
 - Service Discovery, 189
- Erroneous node detection, 101
- Errors
 - in hard real-time systems, 120
 - in Real Time and Resource Overload language, 130
 - SOS, 112
- Ethernet-like communications, 94–95
- EVALUATE PAPERS FAST pattern, 516
- Event and Notification Services, 72
- Event channel dispatching components, 72
- Event-driven communication, 94–95
- Event Driven Invocation, 323
- EVERYONE DOES EVERYTHING pattern, 497
- EVERYONE SPECIALIZES pattern, 497
- Evidence for reuse, 404–407, 425
- EVOCATIVE PATTERN NAME pattern, 434
- Evolution of systems, 6–7
- Evolved systems, pattern languages
 - as, 456–457
- EXCEPTIONAL VALUE pattern, 470–471
 - CROSS LINKAGES, 491
 - RESOLUTION OF FORCES, 482–483
 - VOID, 501
- Expat standard, 291
- Extended Plug-in Contract pattern, 314
- External comparisons, 178–179
- FACADE pattern
 - COMPONENT WRAPPER, 372
 - FRAMEWORK PROVIDING APPLICATION, 317
 - OBJECT SYSTEM LAYER, 359, 374
- Fail silent systems, 120
- Failures in hard real-time systems, 120
- Fallback operations, 166
- FAMILY OF ENTRANCES pattern, 469–470, 481
- FAMILY ROOM CIRCULATION pattern, 498
- Fault containment units (FCUs), 121
- Fault hypotheses, 121
- Fault-tolerance
 - in Real Time and Resource Overload language, 131
 - in safety-critical systems, 92
 - in SYNC FRAME, 110
 - in TIME-TRIGGERED CLOCK SYNCHRONIZATION, 105
- Fault-tolerant units (FTUs), 121
- Faults in hard real-time systems, 120
- FCUs (fault containment units), 121
- FDSs (Fragment Definition Sets), 281–282
- Feedback from shepherds, 509
- FINAL HANDLING pattern
 - in Equitable Resource Allocation, 147
 - in Real Time and Resource Overload language, 131–132, 148
- Financial industry. *See* Web portal framework
- FINDABLE SECTIONS pattern, 516

- FINE-GRAINED OBJECTS pattern,
 - 418FINDER pattern, 30, 33–35
- FINISH WORK IN PROGRESS pattern, 129
 - OVERLOAD EMPIRES, 134
 - Real Time and Resource Overload
 - language, 131, 148
 - REASSESS OVERLOAD DECISION, 136
- Firefox plug-ins, 304
- FIREPLACE AS REFUGE pattern, 541, 543, 546–547
- Fitting Form pattern, 525–526
- Flexibility
 - COMPARAND, 183
 - DYNAMIC OBJECT MODEL, 10
 - in PRESCHEDULED PERIODIC TRANSMISSION, 102
- FlexRay architecture
 - in hard real-time systems, 122–123
 - in PRESCHEDULED PERIODIC TRANSMISSION, 102–103
- FLYWEIGHT pattern, 467, 501
- FOCUS ON LONG-TERM RELEVANCE pattern, 421–422
- Foote, Brian
 - LOW SURFACE-TO-VOLUME RATIO, 418
 - PROTOTYPE A FIRSTPASS DESIGN, 415
 - SELFISH CLASS, 410
- FORCES DEFINE PROBLEM pattern, 436
 - FORCES HINT AT SOLUTION pattern, 446
 - FORM FOLLOWS FUNCTION, 525
 - in shepherding, 508–509, 521–522
- FORCES HINT AT SOLUTION pattern, 434, 444–446
- FORM FOLLOWS FUNCTION pattern, 509, 525–527
- FORM FOLLOWS PREDOMINANT FEATURE pattern, 537–539
 - BANDS OF WINDOWS, 546
 - CANTILEVERED TERRACE, 544
- Forwarding functions, 56
- Fowler, Martin
 - Analysis Patterns*, 22
 - on knowledge level, 5
 - TEMPLATE VIEW, 279
- Frag language, 385, 390–393
- Fragment Definition Sets (FDSs), 281–282
- FRAGMENTS pattern, 257, 260
 - CONTENT CACHE, 264, 285–287
 - CONTENT CONVERTER, 272–273
 - CONTENT FORMAT TEMPLATE, 277–279
 - GENERIC CONTENT FORMAT, 267
 - overview, 262
 - PUBLISHER AND GATHERER, 268
 - Web content conversion and generation, 279–284, 287, 290–291
- Framegrabbers, 89
- Framework development
 - acknowledgments, 427
 - conclusions, 425–426
 - evidence for reuse, 404–407
 - framework user involvement, 420–423
 - introduction, 401–402
 - multiple change requests, 423–425
 - pilot applications, 413–416
 - pilot-based tests, 418–420
 - references, 427–429
 - roadmap, 403–404
 - simplicity, 407–411
 - skilled teams, 411–413
 - small objects, 416–418
- Framework Interface, 311
- FRAMEWORK PROVIDING APPLICATION
 - pattern, 305, 310, 315–319
 - COOPERATING PLUG-INS, 330
 - PLUG-IN BASED PRODUCT, 333
 - PLUG-IN CONTRACT, 313
 - PLUG-IN REGISTRATION, 321

- FRAMEWORK USER INVOLVEMENT pattern, 415, 420–423
- FRESH WORK BEFORE STALE pattern
 - in Equitable Resource Allocation, 146
 - OVERLOAD EMPIRES, 133, 134
 - in Queue for Resources, 143–144
 - in Real Time and Resource Overload language, 131, 149
- Front distance credibility, 99
- FRUIT TREES pattern, 484, 487
- FTUs (fault-tolerant units), 121
- Functional Form pattern, 525–526
- Functors, 55
- Fundamental process, 463
- Fundamental units, 470

- Garbage collection, 79
- GARDEN GROWING WILD pattern, 484, 487, 489
- GARDEN SEAT pattern, 486, 487
- Gaston, K. J., 491
- GASTRULATION pattern, 477–478
- Gatherer class, 288–289
- General Packet Radio Services (GPRS), 213
- GENERIC CONTENT FORMAT pattern, 257, 260
 - FRAGMENTS, 281, 284
 - overview, 261
 - PUBLISHER AND GATHERER, 267–268
 - Web content conversion and generation, 264–267, 287, 289
- Generic representation of Web content, 265
- Generic Security Services (GSS), 348
- Ginko client, 186
- Global Resource Manager (GRM), 351
- Global System for Mobile communications 900 (GSM-900), 213
- Global variables in encapsulation, 50

- GLOBUS grids, 350
- Globus Resource Allocation Manager (GRAM), 350
- GoF book, 526
- GOF pattern form, 437
- Goldchleger, Andrei
 - biographical information, 556
 - GRID, 337
- Golubitsky, M., 453
- “Good For What Ails You” syndrome, 523
- GPRS (General Packet Radio Services), 213
- Grabow, S., 461
- GRAM (Globus Resource Allocation Manager), 350
- Green, R., “Hide Forbidden Globals”, 59
- GREENHOUSE pattern, 484
- Grid clusters, 341–342, 348–349
- Grid Computing, 338–339
- GRID pattern
 - acknowledgments, 353
 - consequences, 351–352
 - context, 339
 - dynamics, 343–344
 - example, 337–339
 - implementation, 343, 345–350
 - intent, 337
 - known uses, 350–351
 - problem, 339–340
 - references, 354–356
 - related patterns, 353
 - solution, 340
 - structure, 340–342
- Grid Security Infrastructure (GSI), 350
- GRM (Global Resource Manager), 351
- Gruber, Manfred
 - biographical information, 556
 - hard real-time systems, 89

- GSI (Grid Security Infrastructure), 350
- GSM-900 (Global System for Mobile communications 900), 213
- GSS (Generic Security Services), 348
- GUIDs, 182
- Haase, Arno
 - biographical information, 557
 - COMPARAND, 169
- HALF A LOAF pattern, 508, 511–512, 514–515
- HALF-HIDDEN GARDEN pattern, 484, 486–487, 489
- HALF-OBJECT PLUS PROTOCOL (HOPP), 459, 476–477
- HALF-SYNC/HALF-ASYNC pattern, 120
- HANDLE/BODY pattern, 465–466
 - AGGREGATION, 496–497
 - COMMON GROUND, 499
 - DIFFERENTIATION, 493
 - LOCAL SYMMETRIES, 474–476
- Handoff procedures in mobile wireless systems, 217–221, 240–242
 - anchor entities in, 242–244
 - failure action for, 246–247
 - intersystem, 244–246
 - summary, 254–255
- Hanmer, Robert S.
 - biographical information, 557
 - Real Time and Resource Overload language, 127
- Hard real-time systems
 - brake-by-wire example, 90–92
 - BUS GUARDIAN, 111–115
 - introduction, 89–90
 - known uses, 122–123
 - patterns-outline, 92–93
 - PRESCHEDULED PERIODIC TRANSMISSION pattern. *See* Prescheduled Periodic Transmission pattern
 - references, 124–125
 - SYNC FRAME, 106–111
 - TEMPORAL APPLICATION DECOUPLING, 115–120
 - terminology, 120–122
 - TIME-TRIGGERED CLOCK SYNCHRONIZATION, 103–106
- Hardware selection for MHP product lines, 386–389
- Harrison, Neil B.
 - advanced pattern writing, 433
 - biographical information, 557
 - “Language of Shepherding”, 436
 - shepherding, 507
- HashMap class, 4
- Hashtables, 14, 20
- Haugen, Robert, 22
- Herzner, Wolfgang
 - biographical information, 557
 - hard real-time systems, 89
- Hidden Globals, 59
- “Hide Forbidden Globals” (Green), 59
- Hierarchical directories, 200
- Hildebrand, G.
 - essential characteristics, 533–534
 - Prairie Houses, 535–536
 - The Wright Space*, 532
- HIP host identities, 202
- HLRs (home location registers)
 - databases for, 236
 - in mobile wireless systems, 215
- Home databases in mobile wireless systems, 220, 225–226, 235–237
 - in paging, 233–234
 - summary, 254
- “Home in a Prairie Town, A”, 531
- Home interface, 39
- Home location areas, 215
- Home location registers (HLRs)
 - databases for, 236
 - in mobile wireless systems, 215

- HOMOGENOUS ADDITION pattern, 465, 490
- HOOK INJECTOR pattern, 384, 391
- Horizontal splits, 53
- Horizontal spread in Prairie Houses, 538–539
- Horizontal tasks, 402–403
- Host Application, 315–319
- Host objects, 350
- Hot spots plug-ins, 316
- How Buildings Learn* (Brand), 463
- “How”-PROCESS pattern, 434, 438–440
- HTML, 258
 - CONTENT CREATOR, 274–277
 - Web content conversion and generation, 288–290
- Htmllib, 292
- Hyperlinks, 450
- I-frames, 111
- ICMPv6 Router Advertisements, 194
- IDENTIFICATION patterns, 203
- Idioms languages, 459, 465
 - AGGREGATION, 496
 - COMMON GROUND, 499
 - CROSS LINKAGES, 489–490
 - DIFFERENTIATION, 493
 - LOCAL SYMMETRIES, 474
 - operator overloading in, 465
- IDL interface, 39
- IEEE 802.11 wireless LAN Beacon messages, 194
- IMT-200 (International Mobile Telecommunications 2000) Systems, 214
- IN THE REGION OF WHOLE VALUE pattern, 467
- Indirection, 32
- Information Integrity, 459
- INFORMATION MARKETPLACE pattern, 422
- Information secrecy, 229, 232
- Inheritance, 12
- Initial Resync, 111
- Input processing in CONTENT CONVERTER, 271
- Insertion collaborations, 30
- Instance class, 9
- Instances, plug-in, 307
- Instantiation in encapsulation, 50, 58
- Insurance, 21, 402–403, 407, 415, 419–422
- InteGrade services, 351
- Integration, component and language.
 - See Component and language integration
- Integrity Controller, 138
- Intended audiences, 447
- Interbase station transceiver handoff, 217
- INTERFACE DESCRIPTION pattern, 360
 - AUTOMATIC TYPE CONVERTER, 377, 378–381
 - COMPONENT WRAPPER, 370
 - SPLIT OBJECT, 393–394, 397
- Internal comparisons, 178–179
- International Mobile Telecommunications 2000 (IMT-200) Systems, 214
- Internet browsers
 - FRAMEWORK PROVIDING APPLICATION, 318
 - PLUG-IN, 309
- Internet Explorer plug-ins, 304
- INTERPRETER pattern, 359–360
 - COMMAND LANGUAGE, 363, 365–367
 - DYNAMIC OBJECT MODEL, 13
 - ENCAPSULATED CONTEXT, 61
 - OBJECT SYSTEM LAYER, 374
 - SPLIT OBJECT, 384, 390, 391, 394–396
- Intersystem handoffs, 218, 220–221, 244–246, 255
- Intrasystem handoffs, 218

- Introduce Parameter Object pattern, 62
- ITERATORS pattern, 501
- IT'S A RELATIONSHIP NOT A SALE pattern, 416
- JAC framework, 367
- Java and Java-like languages
 - comparisons in, 177
 - DYNAMIC OBJECT MODEL, 7
 - OBJECT SYSTEM LAYER, 374
- Java Connector Architecture (JCA), 291
- Java Debug Interface (JDI), 169, 184
- Java Debug Wire Protocol (JDWP), 169
- Java Platform Debugger Architecture (JPDA), 169
- Java RMI (JRMJ), 184
- Java Virtual Machine (JVM), 169–170
- Java Virtual Machine Debug Interface (JVMDI), 169
- JCA (Java Connector Architecture), 291
- JDI (Java Debug Interface), 169, 184
- JDWP (Java Debug Wire Protocol), 169
- Jiffy server, 60
- Jini services
 - CONSULT DIRECTORY, 196
 - SEPARATE IDENTITY FROM LOCATION, 202
 - SERVICES REGISTER IN DIRECTORY, 197–198
- Jitter in hard real-time systems, 121
- Johnson, Ralph
 - biographical information, 558
 - DYNAMIC OBJECT MODEL, 3
 - FINE-GRAINED OBJECTS, 418
 - THREE EXAMPLES, 407
- Join protocols, 197–198
- Jolin, Art, 410
- JPDA (Java Platform Debugger Architecture), 169
- JRMI (Java RMI), 184
- JVM (Java Virtual Machine), 169–170
- JVMDI (Java Virtual Machine Debug Interface), 169
- Kelly, Allan
 - biographical information, 558
 - ENCAPSULATED CONTEXT, 45
- Kerberos security, 348
- Keys
 - DYNAMIC OBJECT MODEL, 14
 - JavaBeans, 185
- Kinds, plug-in, 307
- Knowledge level, 5, 22
- Kon, Fabio
 - biographical information, 558
 - GRID, 337
- Koponen, Teemu
 - biographical information, 558
 - Service Discovery, 189
- Kubinger, Wilfried
 - biographical information, 558
 - hard real-time systems, 89
- Laboratory Systems Manager (LSM), 304, 309
 - COOPERATING PLUG-INS, 329
 - FRAMEWORK PROVIDING APPLICATION, 318
 - PLUG-IN CONTRACT, 315
 - PLUG-IN LIFECYCLE, 324
 - PLUG-IN PACKAGE, 327
 - PLUG-IN REGISTRATION, 321
- Language Designer's Pattern Language (LDPL), 453
 - AGGREGATION, 494–497
 - basis for, 454–457
 - CLUSTER OF FORCES, 467–471
 - COMMON GROUND, 497–500
 - conclusion, 502–503
 - CROSS LINKAGES, 487–491

- DIFFERENTIATION, 491–494
 - examples, 457–459
 - introduction, 453
- LEVELS OF SCALE, 483–487
- LOCAL REPAIR, 461–467
- LOCAL SYMMETRIES, 471–479
- pattern language, 459–461
- references, 502–506
- RESOLUTION OF FORCES, 479–483
- VOID, 500–502
- LANGUAGE EXTENSION pattern, 365
- “Language of Shepherding, The: A Pattern Language for Shepherds and Sheep” (Harrison), 436
- LANs (Local Area Networks), 214
- LAYER pattern, 359
- Layers, 448–449
- LDAP directory, 200
- LDPL. *See* Language Designer’s Pattern Language (LDPL)
- LEASING pattern
 - DECENTRALIZED LOCKING, 165
 - SERVICES REGISTER IN DIRECTORY, 197
- Leaves, HTML, 290
- Legion grids, 350
- LEVELS OF SCALE pattern, 460, 483–487
 - LOCAL REPAIR, 467
 - LOCAL SYMMETRIES, 479
- Libraries
 - GRID, 348
 - OBJECT SYSTEM LAYER, 376
 - SPLIT OBJECT, 385, 393–394, 396
- Liebenau, John
 - biographical information, 558–559
 - DOMAIN OBJECT MANAGER, 25
- LIGHT ON TWO SIDES OF EVERY ROOM pattern, 546
- Limburg, K. E., 486
- Linux plug-ins, 304
- Liskov Substitution Principle (LSP), 49
- LISTEN TO ADVERTISEMENTS pattern, 193–194
 - CONSULT DIRECTORY, 196
 - PLACE DIRECTORIES DYNAMICALLY, 201
 - SERVICES REGISTER IN DIRECTORY, 197
 - in SLP, 206
 - USE ADVERTISER, 195
- Little helpers, 328
- Local Area Networks (LANs), 214
- Local Lock Acquisition scenario, 159–160
- LOCAL-REGIONAL SPECIES RICHNESS RELATIONSHIP pattern, 491
- LOCAL REPAIR pattern, 460
 - AGGREGATION, 497
 - context, 461
 - examples, 465–467
 - forces, 461–462
 - problem, 461
 - rationale, 463–465
 - solution, 462
 - VOID, 502
- LOCAL REPAIR OF THE LANGUAGE pattern, 467
- Local Resource Manager (LRM), 351
- Local scheduling, 346
- LOCAL SYMMETRIES pattern, 460
 - AGGREGATION, 497
 - COMMON GROUND, 500
 - context, 471
 - CROSS LINKAGES, 491
 - DIFFERENTIATION, 494
 - examples, 474–478
 - forces, 472
 - LEVELS OF SCALE, 487
 - LOCAL REPAIR, 467
 - problem, 471
 - rationale, 473–474
 - related patterns, 479
 - solution, 473

- LocalSystem class, 324
- LocalSystemPlugin class, 309–310, 314–315
- Locations in mobile wireless systems, 215, 220, 237–239, 254
- Lock-Permit. *See* DECENTRALIZED LOCKING pattern
- Lock Relay variant, 165
- Lock Release scenario, 159–160
- Locking Overhead force, 157
- LockManager service, 158, 162–165
- LockManager Proxy service, 158–161, 165
- Locks
 - decentralized. *See* DECENTRALIZED LOCKING pattern
 - in dispatching components, 75–78
- Logical design of plug-ins, 308
- Logically deleted objects in reference counting, 80
- LogManager class, 48
- Logrippo, Luigi
 - biographical information, 559
 - MoRaR, 213
- Long Resync frames, 111
- LOOKUP pattern
 - GRID, 353
 - SPLIT OBJECT, 393
- Lookup services, 196
- LOW SURFACE-TO-VOLUME RATIO, 418
- LRM (Local Resource Manager), 351
- LSM. *See* Laboratory Systems Manager (LSM)
- LSP (Liskov Substitution Principle), 49
- MAHO (mobile station-assisted hand-offs), 241
- Maintenance work, 139
- MANAGER pattern, 29
- MANDATORY ELEMENTS PRESENT pattern, 516
- Manolescu, Dragos, 559
- Maps
 - framework development, 409
 - patterns, 508–510
 - Real Time and Resource Overload language, 129–130
 - UCMs, 218
- MarketContext class, 54, 56, 59–60
- MarketDataStore class, 48
- MarketMessageCommand class, 47–48, 54–55, 59
- Marquardt, Klaus
 - biographical information, 559
 - plug-ins, 301
- MARSHALLER pattern, 396
- Mash toolkit, 389
- MASK PRIORITIES TO SHED WORK pattern
 - in Real Time and Resource Overload language, 141–142
 - in Working Hard, Don't Fix it, 140
- MASTER-SLAVE pattern, 353
- MATCHING PROBLEM AND SOLUTION pattern, 436
 - FORM FOLLOWS FUNCTION, 526
 - problem and solution in, 442, 518–519
 - in shepherding, 509
- Matchmaker module, 350
- MDS (Monitoring and Discovery Service), 350
- MEANINGFUL METAPHOR NAME pattern, 435
- MEANINGLESS BEHAVIOR pattern
 - CROSS LINKAGES, 490, 491
 - VOID, 501
- MEDL (Message Description List), 102

- Memory
 - for COMPARAND, 184
 - for OVERLOAD EMPIRES, 133
 - in TEMPORAL APPLICATION DECOUPLING, 119
- MESSAGE INTERCEPTORS pattern
 - COMPONENT WRAPPER, 370
 - CONTENT CONVERTER, 273
 - OBJECT SYSTEM LAYER, 375
 - SPLIT OBJECT, 386
- Message RAM, 119
- MESSAGE REDIRECTOR pattern, 360
 - OBJECT SYSTEM LAYER, 359, 374
 - PUBLISHER AND GATHERER, 268–270
 - SPLIT OBJECT, 386
- Message slots, 98
- Meszaros, Gerard, “Pattern Language for Pattern Writing, A”, 434, 509
- Meta-patterns, 431–432
- Metadata in SPLIT OBJECT, 394
- Methods for States pattern, 61
- MHP (Multimedia Home Platform), 276
 - products, 386–389
 - Web content conversion and generation, 293
- Middleware, 350
- MIND YOUR MANNERS pattern, 441
- Minimum front distance, 99
- Mobile station-assisted handoffs (MAHO), 241
- Mobile stations, 215–218
- Mobile switching centers (MSCs), 215–218
- Mobility and radio resource management. *See* MoRaR pattern language
- Mobility management functions, 221–222
- Module Interface, 119
- Monitor locks, 75–76
- Monitor Object pattern
 - in dispatching components, 75
 - ENCAPSULATED CONTEXT, 62
- Monitoring and Discovery Service (MDS), 350
- Montgomery, Warren, 492–493
- MoRaR pattern language
 - acknowledgments, 250
 - anchor entities, 242–244
 - architectural concepts, 215–218
 - authentication, 215, 220, 225–227, 229–232
 - ciphering, 226–229
 - handoff decisions, 240–242
 - handoff failure actions, 246–247
 - home and visitor databases, 235–237
 - intersystem handoffs, 244–246
 - introduction, 213–214
 - location registration, 237–239
 - mobility management functions, 221–222
 - paging, 233–235
 - pattern language, 218–221
 - radio resource management, 239
 - references, 250–253
 - releasing resources, 247–248
 - security database, 225–226
 - summary, 253–255
 - temporary identification, 222–224
- Motivation, 3
- MPI API, 348, 350
- MSCs (mobile switching centers), 215–218
- Multimedia Home Platform (MHP), 276
 - products, 386–389
 - Web content conversion and generation, 293

- MULTIPLE CHANGE REQUESTS pattern, 410, 413, 423–425
- Multiple objects, dispatching to, 80–85
- Multiple Plug-in Contracts pattern, 314
- Multiple read permits, 167
- Multithreading
 - in CORBA Object Adapter, 72–73
 - ENCAPSULATED CONTEXT, 58
- Mutexes
 - ENCAPSULATED CONTEXT, 58
 - in serialized dispatching, 76
- Naming patterns, 434–436
- Natural systems, 456–457
- Nature of Order* (Alexander), 463, 485
- NetBIOS networks, 201
- Netscape plug-ins, 304
- Network management systems, 155–156
- Network Simulator (NS), 385, 389
- NEURULATION pattern, 478
- Noble, James
 - Arguments Object, 62
 - biographical information, 560
 - OBJECT SYSTEM, 22
- Nodes in hard real-time systems, 121
- Non-existent objects in CORBA, 72
- NOUN PHRASE NAME pattern, 434
- NP-complete problems, 121
- NP-hard problems, 121
- NS (Network Simulator), 385, 389
- NULL OBJECT pattern, 501–502
- Object activation/deactivation use cases, 72
- Object Adapter, CORBA, 71–73
- Object COBOL, 373–374
- Object Factories, 39
- Object ids (OIDs), 32–33
- Object Request Broker (ORB), 69
- OBJECT SYSTEM pattern. *See* DYNAMIC OBJECT MODEL pattern
- OBJECT SYSTEM LAYER pattern, 359–360
 - consequences, 375–376
 - context, 372–373
 - discussion, 374–375
 - forces, 373–374
 - known uses, 376
 - OBJECT SYSTEM LAYER, 374
 - problem, 373
 - scenario, 373, 375
 - solution, 374
 - SPLIT OBJECT, 384, 386, 394, 397
- Objectivity database system, 165–166
- OBSERVER pattern
 - CONSULT DIRECTORY, 196
 - in dispatching components, 73
 - DYNAMIC OBJECT MODEL, 13
 - ENCAPSULATED CONTEXT, 61–62
- OID pattern, 186
- OIDs (object ids), 32–33
- OKBOX pattern, 387
- Olympic Games 2000 Web Site, 293
- One Plug-in per Task pattern, 327
- Opdyke, William, 415
- Open Mash project, 385
- Operational level, 5
- Opportunistic computing, 349
- Oracle system, 166
- ORB (Object Request Broker), 69
- Order class, 26, 36–37
- Order pattern, 30
- Order-processing framework, 26
- OrderFinder interface, 26–28, 30, 37
- OrderManager interface, 26–29, 35–36
- Orders class, 26
- Oregon Experiment, The*, 501
- O’Ryan, Carlos, 69
- Ostwald, J., 457
- OTcl, 385, 389

- OURGRID pattern, 351
- OurGridPeer modules, 351
- Out-of-sync transmissions, 114
- OUTDOOR ROOM pattern, 484
- Output processing in CONTENT CONVERTER, 271
- Overhead
 - in dispatching components, 73
 - in Prescheduled Periodic Transmission, 102
- OVERLAY NETWORK pattern, 192–193
- OVERLOAD ELASTICS pattern
 - OVERLOAD EMPIRES, 134
 - in Real Time and Resource Overload language, 131, 144–145
- OVERLOAD EMPIRES pattern
 - OVERLOAD ELASTICS, 144
 - in Queue for Resources, 143
 - in Real Time and Resource Overload language, 131–134
 - refinement of, 129
- OVERLOAD OUT-OF-CONTROL pattern, 132
- Owner class, 8–9
- Paging
 - in mobile wireless systems, 215, 220, 225, 233–235
 - summary, 254
- Palladio, Andrea, 535
- Parallel computation, 348
- Parameter Block pattern, 62–63
- Parameter Database pattern, 239
- Parameter lists in encapsulation, 50
- Parameterizing Finders, 33
- Parlsberg, Jens, 516
- Parsing Web content, 291
- Parssinen, Juha
 - biographical information, 560
 - Service Discovery, 189
- Partially removed objects in reference counting, 80
- Pattern Almanac, The* (Rising), 448
- Pattern Language, A—Towns, Buildings, Construction* (Alexander), 259
- “Pattern Language for Pattern Writing, A” (Meszaros and Doble), 434, 509
- “Pattern Language for Writers’ Workshops” (Coplien), 508
- Pattern Language of Feature Interaction pattern, 239
- Pattern Languages of Program Design*, 526
- Pattern Writing Patterns, 509
- Patterns
 - advanced. *See* Advanced pattern writing
 - maps of, 508–510
- PCS-1900 (Personal Communication System 1900), 213
- Peer-to-peer file sharing networks, 193
- PEOPLE KNOW BEST pattern, 204
- Performance in locking, 166
- Periodic tasks, 121
- Peripheral equipment for OVERLOAD EMPIRES, 133
- Perl language, 367
- Permit Revoke scenario, 160–161
- Permits
 - in locking, 157
 - multiple read, 167
- PersistenceManager interface, 39
- Persistent states, 28–29, 32–33
- Personal Communication System 1900 (PCS-1900), 213
- Phone Call class, 476–477
- Photoshop program, 304, 308–309, 324
- PHP language, 293
- Physical design for plug-ins, 308

- Physical systems, 457
- PIECEMEAL GROWTH pattern, 487
- PIGGYBACK pattern, 365–366
- PILOT APPLICATIONS pattern, 407, 413–416, 420, 423
- PILOT-BASED TESTS pattern, 416, 418–420
- Pipes, 98
- PLACE DIRECTORIES DYNAMICALLY pattern, 200–201
 - CONSULT DIRECTORY, 196
 - OVERLAY NETWORK, 193
- Platitudes, 444
- PLMN (Public Land Mobile Network), 215
- PLUG-IN pattern, 306–310
- PLUG-IN BASED APPLICATION pattern, 323
- PLUG-IN BASED PRODUCT pattern, 310, 314, 330–333
- PLUG-IN CONTEXT pattern, 315–319
- PLUG-IN CONTRACT pattern, 305, 310–315
- Plug-in Definition interface, 311
- PLUG-IN LIFECYCLE pattern, 305, 322–324
- PLUG-IN PACKAGE pattern, 305, 310, 325–327
- PLUG-IN REGISTRATION pattern, 305, 319–321
- PLUG-IN SUBCONTRACT pattern, 314
- Plug-ins, 305–310
 - acknowledgments, 333
 - vs. components, 302–303
 - COOPERATING PLUG-INS, 327–330
 - example, 303–304
 - FRAMEWORK PROVIDING APPLICATION, 315–319
 - known uses, 304
 - PLUG-IN, 306–310
 - PLUG-IN BASED PRODUCT, 330–333
 - PLUG-IN CONTRACT, 310–315
 - PLUG-IN LIFECYCLE, 322–324
 - PLUG-IN PACKAGE, 325–327
 - PLUG-IN REGISTRATION, 319–321
 - references, 333–335
 - roadmap, 304–306
- POINTERS TO DETAIL pattern, 434, 444, 448–451
- Poltergeist antipattern, 47
- Pools in Equitable Resource Allocation, 147
- Portability in DYNAMIC OBJECT MODEL, 12
- PortalToGo architecture, 291
- POSITIVE CLOSURE pattern, 513
- POSITIVE FEEDBACK FIRST pattern, 513
- POSITIVE OUTDOOR SPACE pattern, 500
- Power consumption in Prescheduled Periodic Transmission, 102
- Power Tools, 309
- Prairie Houses, 531–533
 - acknowledgments, 553
 - Alexandrian rendition, 534–537
 - assessment and conclusion, 551–553
 - BANDS OF WINDOWS, 545–546
 - CANTILEVERED TERRACE, 543–545
 - CHIMNEY AS ANCHOR, 541–543
 - CIRCUITOUS APPROACH, 550–551
 - CONCEALED VERTICALS, 549–550
 - FIREPLACE AS REFUGE, 546–547
 - FORM FOLLOWS PREDOMINANT FEATURE, 537–539
 - PROSPECT AND REFUGE, 533–534, 539–541
 - PROSPECTIVE VIEWS, 547–548
 - references, 553–554
- PrepStmtOrderFinder interface, 30
- PRESCHEDULED PERIODIC TRANSMISSION pattern, 93
 - consequences, 101
 - context, 93
 - example, 93–94, 99–100

- implementation, 97–99
- known uses, 102–103
- problem, 94–96
- related patterns, 103
- solution, 96–97
- Primary keys, 185
- PrimitiveContent class, 266
- Printers, 206–207
- PRIVATE TERRACE ON THE STREET pattern, 484
- Processing pipes, 98
- Processor CPU time, 133
- PROFILE-BASED SERVICE BROWSING pattern, 204
- Programming environment, 11
- PROMOTE AND ADD pattern, 490
- PROMOTION LADDER pattern, 465–466, 490
- Property class, 4–7, 9
- PROPERTY LIST pattern, 4–5, 8–9, 17–19, 22
- PropertyType class, 5–7, 9, 14, 19–20
- PROSPECT AND REFUGE pattern, 539–541
 - BANDS OF WINDOWS, 546
 - CANTILEVERED TERRACE, 545
 - FIREPLACE AS REFUGE, 547
 - PROSPECTIVE VIEWS, 547–548
- PROSPECTIVE VIEWS pattern, 541, 547–548
- PROTOTYPE A FIRSTPASS DESIGN pattern, 415
- Prototype design patterns, 40
- Proven knowledge, 520
- PROXY pattern, 359
 - COMPONENT WRAPPER, 369–371
 - DECENTRALIZED LOCKING, 163–164
- Public Land Mobile Network (PLMN), 215
- Public switched telephone networks (PSTNs), 128
- PUBLISHER AND GATHERER pattern, 257, 260, 264
 - CONTENT CACHE, 285–287
 - CONTENT CONVERTER, 272
 - overview, 261
 - Web content conversion and generation, 267–270, 287
- PUBLISHER-SUBSCRIBER pattern, 196
- PVM API, 348, 350
- Pyarali, Irfan, 69
- Python language
 - COMMAND LANGUAGE, 367
 - OBJECT SYSTEM LAYER, 376
- Quality of Service standards, 130
- Query mechanisms
 - CLIENT KNOWS BEST, 204
 - DOMAIN OBJECT MANAGER, 30, 32
- Queue for Resources
 - OVERLOAD EMPIRES, 133
 - in Real Time and Resource Overload language, 131, 143–144
- Race conditions, 119
- Radio resource management, 217, 239
- RAISED FLOWERS pattern, 484
- Random values in authentication, 230–231
- Rational Rose application, 304, 327
- RdbOrderManager class, 26–28, 30, 36
- RDF library
 - OBJECT SYSTEM LAYER, 376
 - SPLIT OBJECT, 385, 393–394, 396
- Reactive computing systems, 127
- Read permits in locking, 167
- READABLE REFERENCES TO PATTERNS pattern, 436, 450
- Readers/writer locks, dispatching with, 77–78

- READING JUST BEFORE REVIEWING pattern, 516
- Real Time and Resource Overload language
 - acknowledgments, 150–151
 - DISASTER NOTIFICATION, 134–136
 - Equitable Resource Allocation, 146–147
 - introduction, 127–129
 - language context, 130–132
 - language maps, 129–130
 - MASK PRIORITIES TO SHED WORK, 141–142
 - OVERLOAD ELASTICS, 144–145
 - OVERLOAD EMPIRES, 132–134
 - previously published patterns, 148–150
 - Queue for Resources, 143–144
 - REASSESS OVERLOAD DECISION, 136–138
 - references, 151–152
 - Working Hard, Don't Fix it, 138–140
- Real-time systems. *See* Hard real-time systems
- Reasoning in encapsulation, 57
- REASSESS OVERLOAD DECISION pattern
 - OVERLOAD ELASTICS, 145
 - in Real Time and Resource Overload language, 136–138
 - in Working Hard, Don't Fix it, 140
- Recursive access in multithreading, 73
- Recursive applications, 13
- Recursive mutexes, 76
- Redland RDF library
 - OBJECT SYSTEM LAYER, 376
 - SPLIT OBJECT, 385, 393–394, 396
- Redundancy in safety-critical systems, 90
- REFERENCE COUNTER pattern, 499
- Reference counting in dispatch, 78–80
- Reference semantics in comparisons, 171–172
- Refresh rate in scheduling, 97–98
- Regenerative switching delays, 135
- Registering in mobile systems, 215
- Relationship Service Specification, 185
- RELATIONSHIP TO OTHER PATTERNS pattern, 436, 450
- Relationship type objects, 13
- Relationships in shepherding, 517
- Release Strategy variant, 165
- Releasing resources
 - in mobile wireless systems, 220–221, 247–248
 - summary, 255
- Remote interface, 39
- Remote Method Invocation (RMI), 184–185
- remote objects, 183
- Remote procedure call (RPC) protocol, 291
- Remote Proxy, 163
- Remote query interface, 394
- Removal collaborations, 30
- Rendezvous protocol, 192
- Repository pattern. *See* DOMAIN OBJECT MANAGER pattern
- RESOLUTION OF FORCES pattern, 460, 471, 479–483
- RESOURCE LIFECYCLE MANAGER pattern, 353
- Resource management services, 340–341, 346
- Resource module, 350
- Resource providers, 340
- Resource provision services, 340–341, 343–346
- Resource usage information, 346, 352

- Response times in locking, 157
- Reuse
 - COMPARAND attributes, 178
 - in framework development, 404–407, 425
 - GRID, 351
- Revocation requests, 160–163
- Riehle, Dirk
 - biographical information, 560
 - DYNAMIC OBJECT MODEL, 3
 - HALF A LOAF, 515
 - THREE ITERATIONS, 512
- Rising, Linda
 - IT'S A RELATIONSHIP NOT A SALE, 416
 - The Pattern Almanac*, 448
- RMI (Remote Method Invocation), 184–185
- Roaming, 215
- Roberts, Don
 - FINE-GRAINED OBJECTS, 418
 - THREE EXAMPLES, 407
- Round robin scheduling, 145
- RPC (remote procedure call) protocol, 291
- Rule of three, 405–406
- RUNNING EXAMPLE pattern, 525
- Runtime Domain Model. *See* DYNAMIC OBJECT MODEL pattern
- Runtime typing, 12
- Rüping, Andreas
 - biographical information, 560
 - framework development, 401
- Safe Framework Providing Application, 317
- SAFE SETTING pattern, 513
- SAFEbus standard
 - BIUs in, 115
 - in hard real-time systems, 122
 - in SYNC FRAME, 111
 - in TEMPORAL APPLICATION DECOUPLING, 119
- Safety-critical systems, 90, 92
- Salingaros, N. A., 485–486
- Sameness of objects, 172
- Sandboxes in GRID, 347
- Sandboxing Without A Name (SWAN) module, 351
- SavingsAccount class, 15–16
- SavingsAccountType class, 6, 16–17
- SAX standard, 291
- Scalable processor-independent design for electromagnetic resilience (SPIDER), 123
- Scarce resources, 146
- SCATTERED WORK pattern, 488
- Scheduled Invocation plug-ins, 323
- Scheduler objects, 350
- Scheduling
 - round robin, 145
 - services, 340–341, 343, 346–347
 - tasks, 119
- Schmidt, Douglas C.
 - biographical information, 561
 - dispatching components, 69
- Schütz, Dietmar
 - biographical information, 561
 - DECENTRALIZED LOCKING, 155
- Screen savers, 324
- Screening devices, 548
- Script interpreters, 394
- Search engines
 - CONSULT DIRECTORY, 196
 - SERVER DOES HEAVY WORK, 205
- Secrecy, 229, 232
- Secure-channel communication pattern, 232

- Security databases
 - in mobile wireless systems, 220, 225–226
 - summary, 253
- Security in GRID, 340–341, 347–348, 352
- SELECTIVE DYNAMIC OVERLOAD CONTROL pattern
 - in DISASTER NOTIFICATION, 136
 - in Equitable Resource Allocation, 147
 - in Real Time and Resource Overload language, 132, 149
- SELECTIVE TRUNK RESERVATION pattern
 - in Equitable Resource Allocation, 147
 - in Real Time and Resource Overload language, 132, 149
- Self-stabilization in SYNC FRAME, 110
- SELFISH CLASS pattern, 410
- Sell class, 47
- Semantic lookup service, 393–396
- Semi-lattice systems, 488
- Sender authentication pattern, 232
- Senge, P. M., 463–464
- SEPARATE IDENTITY FROM LOCATION pattern, 201–203
 - DIRECTORY FINDS SERVICES, 199
 - in SLP, 206
- Serialized dispatching, 74–76
- Serializer design pattern, 40
- Servants in CORBA, 71
- SERVER DOES HEAVY WORK pattern, 204–205
- Server Lock Acquisition scenario, 159–160
- Server-side caching, 285
- SERVICE ABSTRACTION LAYER pattern, 260, 269–270
- Service centers in locking, 166
- Service discovery, 189–190
 - acknowledgments, 206
 - ALIGN DIRECTORIES WITH ORGANIZATION, 199–200
 - ASK LOCAL NETWORK, 191–192
 - CLIENT KNOWS BEST, 203–204
 - combining patterns, 205–207
 - CONSULT DIRECTORY, 195–196
 - DIRECTORY FINDS SERVICES, 198–199
 - LISTEN TO ADVERTISEMENTS, 193–194
 - OVERLAY NETWORK, 192–193
 - PLACE DIRECTORIES DYNAMICALLY, 200–201
 - references, 206–209
 - SEPARATE IDENTITY FROM LOCATION, 201–203
 - SERVER DOES HEAVY WORK, 204–205
 - SERVICES REGISTER IN DIRECTORY, 196–198
 - USE ADVERTISER, 194–195
- Service Location Protocol (SLP), 190
 - ASK LOCAL NETWORK, 192
 - LISTEN TO ADVERTISEMENTS, 194
 - in service discovery, 205–207
 - SERVICES REGISTER IN DIRECTORY, 197
 - USE ADVERTISER, 195
- Service registration messages, 197
- SERVICES REGISTER IN DIRECTORY pattern, 196–198
 - CONSULT DIRECTORY, 196
 - DIRECTORY FINDS SERVICES, 199
 - SEPARATE IDENTITY FROM LOCATION, 202
 - in SLP, 206
- Session Beans, 61
- Set-top boxes, 386–389
- Severity of overloads, 145
- SHARE THE LOAD pattern, 131, 149

- SHED LOAD pattern
 - OVERLOAD EMPIRES, 133
 - in Real Time and Resource Overload language, 131, 149
 - REASSESS OVERLOAD DECISION, 136–137
 - in Working Hard, Don't Fix it, 139
- SHEPHERD KNOWS THE SHEEP pattern, 508, 512–514
 - Shepherding, 507
 - acknowledgments, 528–529
 - Author as Owner, 517–518
 - BALANCED CONTEXT, 523–524
 - BIG PICTURE, 515–516
 - CONVINCING SOLUTION, 519–521
 - epilogue, 528
 - FORCES DEFINE PROBLEM, 521–522
 - FORM FOLLOWS FUNCTION, 525–526
 - HALF A LOAF, 514–515
 - map of patterns, 508–510
 - references, 529
 - SHEPHERD KNOWS THE SHEEP, 512–514
 - SMALL PATTERNS, 527–528
 - THREE ITERATIONS, 509–512
 - WAR STORIES, 524–525
- “Should” in patterns, 520
- SICO First and Always
 - in Real Time and Resource Overload language, 131–132
 - REASSESS OVERLOAD DECISION, 138
- Simplicity
 - DYNAMIC OBJECT MODEL, 9–10
 - framework development, 407–411
- SimulatorContext class, 60
- Single objects, dispatching to, 74–80
- SINGLE-PASS READABLE PATTERN pattern, 516
- SINGLETON pattern
 - ENCAPSULATED CONTEXT, 61
 - VOID, 501
- SIP URIs, 202
- SIZE THE ORGANIZATION pattern, 413
- SIZE THE SCHEDULE pattern, 413
- SKILLED TEAM pattern, 407, 411–413
- SKIPPABLE SECTIONS pattern, 435, 438
- Slightly out of specification (SOS) errors, 112
- SLP (Service Location Protocol), 190
 - ASK LOCAL NETWORK, 192
 - LISTEN TO ADVERTISEMENTS, 194
 - in service discovery, 205–207
 - SERVICES REGISTER IN DIRECTORY, 197
 - USE ADVERTISER, 195
- Small objects in framework development, 416–418
- SMALL PATTERNS pattern, 509, 527–528
- Smalltalk language
 - class modification in, 7
 - flexibility of, 10
- SOAP standard
 - AUTOMATIC TYPE CONVERTER, 377
 - Web content conversion and generation, 291
- SOB interface, 293
- Soft real-time systems, 89
- Software
 - configurable. *See* Plug-ins
 - integration, 358
- SOS (slightly out of specification) errors, 112
- SPATIAL VARIATION IN ABUNDANCE pattern, 491
- SPECIAL pattern, 501
- SPECIES—AREA RELATIONSHIP pattern, 491
- Specification-based queries, 34–35
- Speed in SYNC FRAME, 110
- SPIDER (scalable processor-independent design for electromagnetic resilience), 123

- SPLIT OBJECT pattern, 359, 382–383
 - Apache Axis, 396–397
 - Aspect configuration, 390–393
 - AUTOMATIC TYPE CONVERTER, 379
 - COMMAND LANGUAGE, 365
 - context, 381
 - discussion, 383–384
 - document archive systems, 386
 - forces, 382
 - OBJECT SYSTEM LAYER, 374
 - problem, 381–382
 - scenario, 382, 384–385
 - semantic lookup service, 393–396
 - set-top boxes, 386–389
 - solution, 382
 - TclCL and XOTcl/SWIG, 389–390
- Sporadic messages, 98
- Sporadic tasks, 121
- Start-up in SYNC FRAME, 110
- State design pattern, 40
- STATE pattern
 - DECENTRALIZED LOCKING, 165
 - VOID, 501
- Statement interface, 30
- STEM CELL SPECIALIZATION pattern, 494
- Stewart, I., 453
- Stock quotes, 280–281, 286
- Storch, D., 491
- StoredProcOrderFinder class, 30, 37–38
- STRATEGIES pattern, 275
- Strategized Locking pattern, 74
- STRATEGY pattern
 - CONTENT CACHE, 286
 - in dispatching components, 74
 - DYNAMIC OBJECT MODEL, 13
 - GRID, 347
 - VOID, 501
- STRING A WIRE pattern
 - in DISASTER NOTIFICATION, 136
 - in Real Time and Resource Overload language, 150
- Structural patterns in MoRaR, 218–221
- Struts framework, 426
- Substitutability in encapsulation, 49, 56
- Substitution rules in XML, 290
- SUNNY PLACE pattern, 484
- SWAN (Sandboxing Without A Name) module, 351
- SWIG wrapper generator
 - COMPONENT WRAPPER, 372
 - SPLIT OBJECT, 389
- Switching delays, 135
- Symmetry breaking, 456–458
- SYNC FRAME pattern
 - consequences, 110
 - context, 106
 - example, 106–107, 109–110
 - in hard real-time systems, 92
 - implementation, 108–109
 - known uses, 111
 - problem, 106
 - related patterns, 111
 - solution, 107–108
- Synchronization
 - in multithreading, 58
 - time-triggered, 101, 103–105
- System costs in BUS GUARDIAN, 114
- System Integrity Control (SICO First and Always)
 - in Real Time and Resource Overload language, 131–132
- REASSESS OVERLOAD DECISION, 138
- T-fault-tolerant midpoint, 105
- TANGENT PATHS pattern, 499
- TARGET READERS pattern, 421
- Tasks
 - application, 116–117
 - in hard real-time systems, 121
 - in TEMPORAL APPLICATION DECOUPLING, 119

- Tautologies, 440
- Taylor, Paul R.
 - biographical information, 561
 - Prairie Houses, 531
- Tcl language
 - AUTOMATIC TYPE CONVERTER, 381
 - COMMAND LANGUAGE, 367
- Tclcl language, 385, 389
- TclHttpd, 294
- TDMA (Time Division Multiple Access)
 - in hard real-time systems, 122
 - in MoRaR, 222
 - in Prescheduled Periodic Transmission. *See* PRESCHEDULED PERIODIC TRANSMISSION pattern
- Telecommunications Input Output Language, 136
- TEMPLATE METHODS pattern, 272
- TEMPLATE VIEW pattern, 279
- TEMPLATES pattern, 273
- TEMPORAL APPLICATION DECOUPLING pattern
 - acknowledgements, 120
 - consequences, 119
 - context, 115
 - example, 116
 - in hard real-time systems, 93
 - implementation, 118–119
 - known uses, 119
 - problem, 115–116
 - related patterns, 120
 - solution, 116–118
- Temporal splits, 53
- Temporary identification
 - in mobile wireless systems, 220, 222–224
 - summary, 253
- Temporary Mobile Subscriber Identity (TMSI), 224
- TERMINOLOGY TAILORED TO AUDIENCE pattern, 436, 447
- Testing
 - ENCAPSULATED CONTEXT, 58
 - GRID, 352
- This pointers, 55
- Thread-Specific Storage pattern, 82
- THREE EXAMPLES pattern, 407
- THREE ITERATIONS pattern, 436, 442, 508–512
- Thumbnails, 450
- Tilman, Michel
 - biographical information, 561
 - DYNAMIC OBJECT MODEL, 3
- TIME DIVISION MULTIPLE ACCESS (TDMA)
 - in hard real-time systems, 122
 - in MoRaR, 222
 - in Prescheduled Periodic Transmission. *See* PRESCHEDULED PERIODIC TRANSMISSION pattern
- TIME-TRIGGERED CLOCK SYNCHRONIZATION pattern, 103–106
 - in hard real-time systems, 92
 - vs. PRESCHEDULED PERIODIC TRANSMISSION, 101–102
- Time-triggered communication on CAN (TTCAN) protocol
 - in hard real-time systems, 123
 - in PRESCHEDULED PERIODIC TRANSMISSION, 102–103
 - in TEMPORAL APPLICATION DECOUPLING, 119
- Time-Triggered Communication pattern. *See* PRESCHEDULED PERIODIC TRANSMISSION pattern
- Time-Triggered Protocol (TTP)
 - in hard real-time systems, 122
 - in Prescheduled Periodic Transmission, 102
- Titles of pattern collections, 447
- TMSI (Temporary Mobile Subscriber Identity), 224

- Token Ring networks, 96
- TPMHP project, 293
- TradingContext class, 60
- TradingDayChange class, 47
- Transient states, 28
- Transmissions. *See* PRESCHEDULED PERIODIC TRANSMISSION pattern
- TREE PLACES pattern, 484, 487, 489
- Trigger-based registration, 321
- Trigger events
 - for location registration, 239
 - for plug-ins, 323
- Trigger memory, 119
- Triple-T system. *See* Hard real-time systems
- TTCAN (time-triggered communication on CAN) protocol
 - in hard real-time systems, 123
 - in PRESCHEDULED PERIODIC TRANSMISSION, 102–103
 - in TEMPORAL APPLICATION DECOUPLING, 119
- TTP (Time-Triggered Protocol)
 - in hard real-time systems, 122
 - in PRESCHEDULED PERIODIC TRANSMISSION, 102
- Two-pass reads, 450
- Twombly, Robert, 532
- Type-checking property access, 14
- TYPE OBJECTS pattern
 - AUTOMATIC TYPE CONVERTER, 379
 - DYNAMIC OBJECT MODEL, 4, 8–9, 22
 - OBJECT SYSTEM LAYER, 375
- Types, plug-in, 307
- UCMs (Use Case Maps), 218
- UDDI protocol
 - SERVER DOES HEAVY WORK, 205
 - SERVICES REGISTER IN DIRECTORY, 198
- UML
 - collaboration, 7–8
 - virtual machines, 21
- Uncluttered code, 58
- Universal Mobile Telecommunications System (UMTS), 213
- Update collaborations, 30
- UPnP protocol
 - ASK LOCAL NETWORK, 192
 - CLIENT KNOWS BEST, 204
 - LISTEN TO ADVERTISEMENTS, 194
 - SEPARATE IDENTITY FROM LOCATION, 202
- URLs in service discovery, 202
- Usage patterns in GRID, 349
- USE ADVERTISER pattern, 194–195
- Use Case Maps (UCMs), 218
- UserAgent module, 351
- Users in framework user involvement, 421
- Validating DomainObjects, 33
- Validity time spans
 - in hard real-time systems, 122
 - in Prescheduled Periodic Transmission, 97
- Value class, 9
- VALUE HOLDER pattern, 4, 9
- Value semantics, 171
- Variable dispatching times, 73
- VCF, 372
- VEGETABLE GARDEN pattern, 484
- Vertical height in Prairie Houses, 538
- Vertical splits, 53
- VISIBLE FORCES pattern, 435, 446, 522
- VISIBLE IMPLICATION pattern, 487
- Visited areas in mobile systems, 215

- Visitor databases
 - in mobile wireless systems, 215, 220, 223, 225–226, 235–237
 - in paging, 233–234
 - summary, 254
- Visitor design pattern, 40
- Visitor location registers (VLRs), 236
- Visual Basic pattern, 11
- Voelter, Markus, 562
- Vogel, Oliver
 - biographical information, 562
 - Web content conversion and generation, 257
- VOID pattern, 461, 500–502
 - AGGREGATION, 497
 - DIFFERENTIATION, 494
 - LEVELS OF SCALE, 487
- VxWorks plug-ins, 304
- Waiting period in SYNC FRAME, 109
- WANs (Wide-Area Networks), 214
- WAR STORIES pattern, 509, 524–525
- WCET (Worst Case Execution Time)
 - in EMB systems, 99
 - in hard real-time systems, 122
- Weasels, 434, 443–444
- Weather forecasting, 337–339
- Web-based applications, 258–259
- Web browsers
 - FRAMEWORK PROVIDING APPLICATION, 318
 - PLUG-IN, 309
- Web content conversion and generation, 257
 - acknowledgments, 295
 - conclusion, 294–295
 - CONTENT CACHE, 284–287
 - CONTENT CONVERTER, 270–274
 - CONTENT CREATOR, 274–277
 - CONTENT FORMAT TEMPLATE, 277–279
 - form, 259–260
 - FRAGMENTS, 279–284
 - GENERIC CONTENT FORMAT, 264–267
 - implementation example, 287–291
 - intended audience, 259
 - introduction, 257–259
 - known uses and related work, 291–294
 - overview, 260–264
 - PUBLISHER AND GATHERER, 267–270
 - references, 295–297
- Web portal framework
 - background, 403
 - evidence for reuse, 407
 - framework user involvement, 422–423
 - multiple change requests, 425
 - pilot applications, 415
 - pilot-based tests, 420
 - simplicity, 410
 - skilled teams, 412–413
 - small objects, 418
- Web search engines
 - CONSULT DIRECTORY, 196
 - SERVER DOES HEAVY WORK, 205
- WebLogic application servers, 39
- WebShell, 293
- WebSphere, 39
- Welch-Lynch algorithm, 105
- “WHAT”—SOLUTIONS pattern, 434
- WHOLE VALUE pattern, 466–467
 - CLUSTER OF FORCES, 470–471
 - CROSS LINKAGES, 491
 - HOPP, 476–477
 - LEVELS OF SCALE, 487
 - RESOLUTION OF FORCES, 482–483
- “WHY”—PROBLEMS pattern, 434, 440–443
- Wide Area Networks (WANs), 214
- Wild clock readings, 104

- WIN (Wireless Intelligent Network), 214
- Windows (operating system) plug-ins, 304, 309
- Windows (Prairie Houses), 545–546
- Winn, Tiffany
 - biographical information, 562
 - LDPL, 453
- Wireless Intelligent Network (WIN), 214
- Wireless mobile Asynchronous Transfer Mode (WmATM) systems, 214, 241
- Wireless systems. *See* MoRaR pattern language
- Witthawaskul, W., 21
- Word application, 304
 - FRAMEWORK PROVIDING APPLICATION, 318
 - PLUG-IN LIFECYCLE, 324
 - PLUG-IN PACKAGE, 327
- WORK SHED AT PERIPHERY pattern, 150
- Worker objects, 55–56
- Working Hard, Don't Fix it strategy, 138–140
- Workloads in real time systems, 128
- Workshops
 - disadvantages of, 446–447
 - framework user involvement, 421
- World Peace syndrome, 519–520
- Worst Case Execution Time (WCET)
 - in EMB systems, 99
 - in hard real-time systems, 122
- WRAPPER FACADE pattern, 359, 370–371
- Wrappers in SPLIT OBJECT, 382–383, 386, 394
- Wright, Frank Lloyd. *See* Prairie Houses
- Wright Space, The—Pattern and meaning in Frank Lloyd Wright's Houses* (Hildebrand), 532
- Writer locks, dispatching with, 77–78
- WSDL files, 394, 397
- X-frames, 111
- XML
 - CONTENT CREATOR, 274–277
 - Web content conversion and generation, 288–290
- XML Database Server, 60–61
- XML Schema Data (XSD), 396–397
- xoComm architecture, 291–292
- xoRDF architecture, 292
- XOTcl language, 376, 389
- XSD (XML Schema Data), 396–397
- XSLT standard, 291
- Yoder, Joseph
 - “Connecting Business Object to Relational Databases”, 40
 - LOW SURFACE-TO-VOLUME RATIO, 418
 - SELFISH CLASS, 410
- Zdun, Uwe
 - biographical information, 562
 - component and language integration, 357
 - Web content conversion and generation, 257
- Zeus, 304
 - COOPERATING PLUG-INS, 329
 - PLUG-IN, 309
 - PLUG-IN BASED PRODUCT, 333
 - PLUG-IN CONTRACT, 315
 - PLUG-IN LIFECYCLE, 324
 - PLUG-IN PACKAGE, 327
 - PLUG-IN REGISTRATION, 321
- Zhao, L., 457
- Zope system, 293