

# INDEX

---

## A

Abstraction, in models, 167–168

Access, authorizing, 382–383

Access control policy, 374

Access control sets, 373

Accountability, security, 369, 384–385

Acquirers, 111, 112

Administration models. *See also* Operational viewpoint.

activities, 340–341

custom utilities, 341

error conditions, 339–341

example, 342–343

monitoring facilities, 339, 341

notation, 340

overview, 339–340

performance metrics, 341

performance monitoring, 340

performance scenarios, 341

routine maintenance requirements, 340–341

routine procedure requirements, 339

Age concerns, 245–246

Aggregation, 254

Agile development methods, 88–89

Agile modeling (AM), models, 171

Architects

*versus*

business analysts, 64

design authorities, 64–66

developers, 66

project managers, 64

technology specialists, 65–66

core concepts, 62

in different projects

enterprise application integration (EAI), 514–515

extending existing systems, 515–516

in-house development, 513

new product development, 513–514

package implementation, 516

domain, 63

enterprise, 63

organizational context, 64–66

product, 63

in project lifecycle

architecture definition, 509–510

decommissioning, 512–513

operation, 511–512

project initiation, 507–508

requirements definition, 509

scope definition, 509

stakeholder identification, 508

system construction, 510

system testing, 510

user acceptance testing, 510

responsibilities, 67

role of, 60–61

skill set, 66–67

solution, 63

specializations, 63

Architectural acceptance criteria, 75

Architectural description (AD)

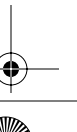
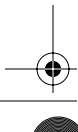
contents

appendices, 187–188

concerns overview, 185

document control, 184

general principles, 185–186





- contents, *continued*
  - introduction, 184
  - management summary, 184
  - overview, 183–184
  - quality property summary, 187
  - requirements overview, 185
  - scenarios, 187
  - scope definition, 185
  - views, 186
- definition, 175
- example, 23
- glossaries, 181–182
- IEEE Standard 1471, 182–183
- introduction, 22–24
- problems, 303
- properties of
  - clarity, 179–180
  - conciseness, 178–179
  - correctness, 176–177
  - currency, 180
  - precision, 180–181
  - sufficiency, 177–178
- styles in, 144–145
- Architectural elements. *See* elements.
- Architectural evaluation, scenario-based, 199–204
- Architectural perspectives. *See* perspectives.
- Architectural requirements, 99
- Architectural significance, definition, 59–60
- Architectural styles. *See* styles.
- Architectural views. *See* views.
- Architecture
  - candidate architecture, 17
  - core concepts, 24
  - definition, 12
  - dynamic structures, 12–13, 15–16
  - external properties
    - externally visible behavior, 13, 17
    - internal organization, 13–14, 17
    - quality properties, 13–14
  - importance of, 17–18
  - intrinsic nature of, 18
  - possible solutions. *See* candidate architecture.
  - static structures, 12, 15–16
  - system structures, 12–13
- Architecture definition
  - activities, 80–85
  - architect's role in, 509–510
  - architectural options, 75, 80–85
  - candidate architecture, 80–85
  - capturing first-cut concerns, 79
  - capturing stakeholder needs, 56–57
  - consolidating inputs, 80–85
  - core concepts, 62
  - defining the architecture, 79
  - definition, 56
  - design inputs, creating, 75
  - exit criteria, 85–86
  - guiding principles, 73–74
  - identifying architectural styles, 80–85
  - identifying scenarios, 80–85
  - initial scope and context, defining, 78
  - inputs/outputs, 77–80
  - iterative development approaches, 88
  - outcomes, 74–75
  - pragmatism, 74
  - process context, 75–76
  - process description, 56–57
  - process outcomes, 74–75
  - requirements
    - analysis, 57, 58–59, 75–76
    - clarifying, 74
    - revisiting, 80–85
  - reworking architecture, 80–85
  - scope, 58
  - skeleton systems, creating, 80
  - in software development lifecycle, 86–89
  - stakeholders
    - concerns, 73
    - engaging, 79
    - evaluating architecture with, 80–85
    - managing expectations, 74–75
  - structure, 74
  - supporting activities, 77–80
  - technological neutrality, 74

- Architecture description languages (ADLs), 164–165
- Architecture design. *See* architecture definition.
- Architecture models. *See* models.
- Architecture Tradeoff Analysis Method (ATAM), 195, 199–204.
- Archive concerns, 250–251
- Assessors, 111, 113
- Asset protection, 497
- Associations, 252–253
- Asynchronous processing, 419
- ATAM. *See* Architecture Tradeoff Analysis Method
- Attack trees, 375–376
- Attributes, 252
- Auditing sensitive events, 381–382
- Authenticating principals, 382
- Authorizing access, 382–383
- Availability
  - functional, 438–439, 441
  - platform, 434–438
  - requirements for, 432–433
  - schedule, 433–434
  - security aspects, 369, 385–386
  - software, 444
  - technology, 441
- B**
- Backout strategy, 334, 336, 349
- Backup. *See also* disaster recovery.
  - concerns, 330–332
  - models, 351
  - solutions, 445–446
- Books and publications
  - Analysis Patterns*, 159
  - Architectural Blueprints: The 4+1 View Model of Software Architecture*, 30
  - IEEE Standard 1471, 182
  - Pattern-Oriented Software Design*, 138
  - Recommended Practice for Architectural Description*, 182
- Boxes-and-lines diagrams, 222–223, 224
- Build process, 302
- Build process, automating, 466–467
- Business analysts *versus* architects, 64
- Business goals and drivers, architectural limits and constraints, 92–93
- C**
- Candidate architecture, 17, 80–85
- Cardinality, 252
- Change. *See* Evolution perspective.
- Class of service, concerns, 428–430
- Classes
  - definition, 252–253
  - of incidents, 344
  - of stakeholders, 111–115
- Client nodes, 311
- Client/server approach, definition, 15
- Code. *See* software.
- Codeline model
  - activities, 302
  - build approach, 302
  - configuration management, 302
  - notation, 302
  - overview, 301–302
  - release process, 302
  - source code structure, 302
- Codeline organization concerns, 295
- Cohesion, Functional viewpoint, 217
- Committing updates, 248
- Common design model
  - activities, 301
  - common processing, 298, 301
  - common software, 299
  - design constraints, 299–300, 301
  - design patterns, 301
  - notation, 299
  - overview, 298–299
  - standard design approaches, 299
  - standard software elements, 301
- Common processing concerns, 294
- Communicating with stakeholders, scenarios, 123
- Communication, via architecture definition, 74



- Communicators, 111, 113
- Compartmentalization security principle, 380
- Complexity
- Concurrency viewpoint, 288
  - Evolution perspective, 457
  - interface, 265–266
  - managing with views and viewpoints, 33
- Compression, 227
- Concerns. *See also specific concerns; specific viewpoints.*
- architectural limits and constraints
    - architectural requirements, 99
    - characteristics of, 99–100
    - definition, 97
    - functional requirements, 98–99
    - goals, 98  - capturing first-cut, 79
  - definition, 20–21
  - mean time to repair (MTTR), 431
  - separating, 32–33
  - stakeholders', 73
- Concurrency-related contention, 423
- Configuration, system
- change strategy, 337–338
  - group dependencies, 337
  - groups, identifying, 337
  - value sets, 337
- Configuration management, software, 302, 329.
- Configuration management models, system
- activities, 337–339
  - configuration change strategy, 337–338
  - configuration groups, 337
  - configuration value sets, 337
  - example, 338–339
  - notation, 337
  - overview, 336–337
- Conflict resolution strategies for data update, 259
- Connectors, functional structure model, 219
- Consistency across views. *See* views, consistency across.
- Constraints. *See* limits and constraints.
- Content equivalence, 477
- Contention and concurrency, 282, 413–414
- Context, defining, 78
- Context diagrams, 95–96
- Corporate asset protection, 497
- Coupling, 217
- Cross-cutting concerns, 5–6
- D**
- Data. *See also* Information viewpoint.
- and control flows, 216
  - incompatibilities, 262–263
  - migration, 327–329, 336
  - ownership, 243–245, 258–260
  - protection, 496
  - quality, 248–249, 263–264
  - quality analysis models, 260–261
  - retention, 250–251, 496
  - self-describing, 465
  - volume, 249–250, 262
- Data-oriented approaches, functional structure model, 229
- Data-sharing mechanisms, 276
- De facto standards, 105–106
- Deadlocks, 281, 289
- Decommissioning, architect's role in, 512–513
- Decomposition, functional structure model, 227
- Defense in depth, 380
- Dependencies
- functional structure model, 238
  - identifying, 298, 333
  - problems, 320–321
- Descoping, 483
- Design
- versus* architecture definition, 57, 59–60
  - constraints, 299–300, 301
  - for failure, 388–389





- inputs, creating, 75
- patterns. *See* patterns.
- standard approaches, 299
- standardization concerns, 294–295
- Design authorities *versus* architects, 64–66
- Design by Contract, 228.
- Detection
  - errors, 447–448
  - intrusions, 369, 378
- Developers, 111, 113
- Developers *versus* architects, *versus*, 66
- Development environments, preserving, 467–468
- Diagrams, functional structure model, 236
- Disability antidiscrimination, 497
- Disabled users. *See* Accessibility perspective.
- Disaster recovery. *See also* backup.
  - activities, 378
  - concerns, 369, 431–432
  - incident recovery analysis, 437–438
  - solutions, 445–446
- Domain analysis, 254
- Domain architects, 63
- Downtime, 430
- Dynamic structures, 12–13, 15–16
- E**
- Elements. *See also* modules.
  - boundaries, 18
  - definition, 18
  - interfaces, 18
  - key attributes, 18–19
  - mapping to elements, 312
  - mapping to tasks, 272, 280
  - physical distance between. *See* Location perspective.
  - replaceable, 465
  - responsibilities, 18
- Elements, mapping to hardware, 314. *See also* elements; modules.
- Encapsulation, to contain change, 461
- Enterprise application integration (EAI), architect's role in, 514–515
- Enterprise architects, 63
- Entities
  - definition, 252
  - external, 219
  - as stakeholders, 19–20
  - state, 282
  - in state machines, 282–283
  - trusting external, 381
- Entity life histories, 257
- Entity-relationship modeling, 252
- Environment (system)
  - invalid assumptions, 422
  - problems, 304–305
  - release management, 467
- Environment (ecology) impact, 497
- Error conditions, 339–341
- Error detection problems, 447–448
- Error handling. *See* Availability and Resilience perspective.
- Escalation process, 344, 345
- Evaluating architecture, scenarios, 122–123
- Events, auditing, 381–382
- Execution coordination mechanisms, 276
- Extending existing systems, architect's role in, 515–516
- External entities, 219
- External interfaces. *See* interfaces, external.
- External properties
  - example, 14–16
  - externally visible behavior, 13, 17
  - internal organization, 13–14, 17
  - quality properties, 13–14
- Externally visible behavior, 13, 17
- F**
- Failure scenarios, 132
- Fault tolerance
  - hardware, 441–442
  - software, 445
- Financial issues, 496





Fitness for purpose, 483  
Flexibility analysis, functional structure model, 232  
Fragmentation, views and viewpoints, 34–35  
Functional availability, 438–439, 441  
Functional capabilities, 216  
Functional elements. *See* Functional viewpoint; Functional views.  
Functional migration concerns, 326–327  
Functional requirements, 41, 98–99  
Functional scenarios, 122  
Functional structure model. *See* Functional viewpoint, functional structure model.

**G**

Generalization, 227  
Groupings, viewpoints, 35–36  
Groups, as stakeholders, 19–20  
Guidelines, architectural limits and constraints, 106

**H**

Handicapped users. *See* Accessibility perspective.  
Hardware  
  clustering, 442–443  
  mapping components to, 314  
  online/offline storage, 312  
  requirements, 308–309, 314–315, 403  
Health and safety, 497

**I**

Idioms, programming language, 138–140, 152–153  
IEEE Standard 1471, 19, 25, 30, 31, 37, 182–183, 188–189  
Incident recovery analysis, 437–438  
Inconsistencies, views and viewpoints, 34. *See also* views, consistency across.  
Indirection, 422–423

**Information**

  flow concerns, 243  
  flow models, 255–257  
  integrity, 384  
  latency, 265  
  lifecycle models, 257–258  
  secrecy, 383–384  
  structure and content concerns, 242–243  
  synchronization approach, 336  
Infrastructure modeling, functional structure model, 233  
In-house development, architect's role in, 513

**Installation concerns, 326**

**Installation model**  
  activities, 333–335  
  backout strategy, 334  
  constraints, identifying, 333–334  
  dependencies, identifying, 333  
  example, 334–335  
  installation groups, identifying, 333  
  notation, 333  
  overview, 332–333

**Instrumentation concerns, 295**

**Integration**

  architectural principle, 74  
  continuous, 467  
  hubs, 265–267  
  with production environment, 350–351  
  security technologies, 386

**Integrity, 273, 373**

**Interaction analysis, functional structure model, 231–232**

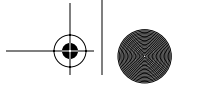
**Interfaces, system**

  complexity, 265–266  
  definition languages, 229  
  design, 232  
  external, Functional viewpoint, 216  
  flexibility, 462–463  
  functional structure model  
    data-oriented approaches, 229  
    definition, 218  
    definition languages, 229



- Design by Contract, 228
- design notations, 229
- designing, 228–231
- programming languages, 228–229
- Interfaces, user. *See* user interface.
- Internal organization, 13–14, 17
- Interprocess communication, 272, 276–277
- Iterative development approaches, 88
- K**
- Key matching, 264–265
- Kruchten, Philippe, 30–31, 59, 122.
- L**
- Language idioms. *See* idioms.
- Latency concerns, data, 245–246
- Law enforcement, 497
- Layering rules, identifying, 298
- Legal issues. *See* Regulation perspective.
- Lifecycles
  - information, 257–258
  - project, architect's role in
    - architecture definition, 509–510
    - decommissioning, 512–513
    - operation, 511–512
    - project initiation, 507–508
    - requirements definition, 509
    - scope definition, 509
    - stakeholder identification, 508
    - system construction, 510
    - system testing, 510
    - user acceptance testing, 510
  - software, 86–89, 204–206
- Lightweight processes, 276
- Limits and constraints
  - business goals and drivers, 92–93
  - concerns
    - architectural requirements, 99
    - characteristics of, 99–100
    - definition, 97
    - functional requirements, 98–99
    - goals, 98
  - constraints assessment, 315
  - context diagrams, 95–96
  - guidelines, 106
  - identifying constraints, 333–334
  - policies, 106
  - principles
    - characteristics of, 101–102
    - defining, 102–103
    - definition, 100
    - demonstrating traceability, 103–105
    - scope, 92, 93–94
    - standards, 105–106
    - strategies, 106
- Load balancing, 442–443
- Localization. *See* International perspective.
- Logging transactions, 443–444
- M**
- Maintainers, 111, 113
- Mapping
  - components to hardware, 314
  - elements to elements, 312
  - elements to tasks, 272, 280
- Mean time before fault (MTBF), 435–436
- Mean time to repair (MTTR), 435–436
- Mechanisms, security, 368–369
- Message interactions, functional structure model, 225–226
- Metadata models, 261
- Meyer, Bertrand, 228
- Migration
  - backout strategy, 336
  - big bang, 326
  - data, approach, 336
  - data, concerns, 327–329
  - functional, concerns, 326–327
  - information synchronization approach, 336
  - parallel run, 327
  - planning, 349
  - problems and pitfalls, 349–350
  - staged, 327
  - strategies, 336
  - window, 349–350



**Migration models**

- activities, 336
- backout strategy, 336
- data migration approach, 336
- information synchronization approach, 336
- notation, 335
- overview, 335
- strategies, 336

**Modeling languages, 164–165**

**Models.** *See also specific models; specific viewpoints.*

- agile modeling (AM), 171
- definition, 157
- Functional viewpoint. *See* Functional viewpoint, functional structure model.
- guidelines for creating
  - abstraction, 167–168
  - change control, 170–171
  - choosing a notation, 170
  - defining a purpose, 166
  - defining terms, 169
  - identifying the audience, 166–167
  - keeping it simple, 169
  - naming, 168–169
  - validation, 170
- importance of, 158–161
- intermodel relationships, 319–320
- qualitative, 161–162
- quantitative, 162–163
- realism problems, 421
- sketches, 164, 223
- types of, 161–164

**Module structure model**

- activities, 298
- classifying modules, 298
- dependencies, identifying, 298
- layering rules, identifying, 298
- notations, 296–297
- overview, 296

**Modules.** *See also* elements.

- classifying, 298
- organization concerns, 294

**Monitoring**

- facilities for, 339, 341
  - operations, 329
  - performance, 330, 340
- Multiple updaters, 264**

**N****Network models**

- activities, 317
- designing the network, 317
- estimating capacity, 317
- network connections, 316
- network nodes, 316
- notation, 317
- overview, 315–316
- processing nodes, 315

**Networks**

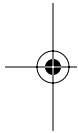
- capacity concerns, 310
- connections, 316
- designing, 317
- estimating capacity, 317
- in-process differences, 424–425
- links, 312
- nodes, 316
- processing nodes, 315
- requirement concerns, 309–310

**New product development, architect's**

- role in, 513–514

**Nonfunctional requirements, 41****Normalization, 253****Notations**

- architecture description language (ADL), 164–165
- boxes-and-lines diagrams, 222–223, 224
- choosing, 170
- functional structure model, 219–226
- graphic modeling, 169
- modeling languages, 164–165
- sketch, 223
- unified modeling language (UML), 165

**O****Object IDs, 246–247****Obscurity, as security measure, 381**



- Off-the-shelf deployment project, 115–116
  - Online/offline storage hardware, 312
  - Open standards, 105–106
  - Operation, architect's role in, 511–512
  - Operational monitoring and control concerns, 329
  - Operational service levels, defining, 433
  - Operational staff, engaging, 348–349
  - Organizational standards, 106
  - Overloaded views, 234–235
- P**
- Package implementation, architect's role in, 516
  - Paper models, 128–129. *See also* scenarios.
  - Parallelizing, 418–419
  - Parameterizing operations, 465
  - Partitioning, 418–419, 422
  - Partnered development project, 117
  - Patterns
    - identifying and defining, 301
    - idioms, 138–140, 152–153
    - organization, 137–138
    - required content, 135–136
    - styles
      - and architectural description (AD), 144–145
      - Asynchronous Data Replication, 149
      - benefits of, 142–144
      - Client/Server, 146
      - definition, 138
      - Distribution Tree, 149–150
      - Integration Hub, 150–151
      - Layered Implementation, 147–148
      - Peer-to-Peer, 147
      - Pipes and Filters, 145–146
      - Publisher/Subscriber, 148–149
      - Tiered Computing, 146–147
      - Tuple Space, 151
      - types of, 145–151
    - versus* tactics, 43
    - uses for, 135–136
  - Peak load behavior, concerns, 403–404
  - People, as stakeholders, 19–20
  - Performance
    - benchmark test, 410–411
    - estimating, 410
    - metrics
      - estimating, 407–408, 410
      - identifying, 341
      - measuring, 410
    - models, 406–410
    - monitoring, 330, 340
    - requirements, capturing, 404–406
    - scenarios, 341
  - Performance-critical structures, 407
  - Perry, Dewayne, 30.
  - Personnel. *See* stakeholders.
  - Perspective catalog, 363–364
  - Perspectives. *See also specific perspectives.*
    - applying to views
      - artifacts of, 48
      - consequences of, 47–49
      - definition, 42
      - description, 44–47
      - improvements from, 48
      - insights from, 47
    - benefits of, 50
    - core concepts, 49
    - definition, 6, 41
    - example, 1–5
    - introduction, 1–6
    - overview, 363–364
    - pitfalls, 50–51
    - quality properties, 39–41
    - standard structure, 43
    - uses for, 42–43
  - Physical constraint concerns, 310
  - Piecemeal security, 392–393
  - Piloting, 483. *See also* prototypes; scenarios; simulations.
  - Platform availability, estimating, 434–438
  - Policies, architectural limits and constraints, 106



- Policies, security, 367–368  
Predictability, concerns, 402–403  
Presentations, architectural validation, 193–194, 204–206  
Primary keys, 246–247  
Principal classes, security, 373  
Principals, security, 366, 382  
Principles  
  architectural description (AD), 23  
  architectural limits and constraints  
    characteristics of, 101–102  
    defining, 102–103  
    definition, 100  
    demonstrating traceability, 103–105  
  architecture, and stakeholder needs, 21  
  architecture definition, 85  
  good architecture definitions, 57  
  intrinsic nature of architecture, 18  
  limits of planning, 29  
  models, 159  
  nature of good architecture, 22  
  requirements analysis, 59  
  responsibilities of architects, 60  
  stakeholder characteristics, 110  
  stakeholder representation, 112  
Prioritizing scenarios, 123–124  
Privileges, granting, 380  
Problems and pitfalls. *See specific problems; specific viewpoints.*  
Procedure call mechanisms, 276  
Process groups, 275–276  
Processes  
  definition, 275  
  lightweight, 276  
  prioritizing, 281  
  reducing to steps, 465–466  
Processing  
  asynchronous, 419  
  common approaches, 298, 301  
  distributing over time, 416–417  
  generic, 465  
  optimizing repeated, 412–413  
  prioritizing, 414–416  
  separating physical from logical, 465  
Processing nodes, 311  
Product architects, 63  
Programming languages, functional structure model, 228–229  
Programs. *See* software.  
Project initiation, architect's role in, 507–508  
Project lifecycle, architect's role in  
  architecture definition, 509–510  
  decommissioning, 512–513  
  operation, 511–512  
  project initiation, 507–508  
  requirements definition, 509  
  scope definition, 509  
  stakeholder identification, 508  
  system construction, 510  
  system testing, 510  
  user acceptance testing, 510  
Project managers *versus* architects, 64  
Proof-of-concept systems, 197–198.  
  *See also* validation.  
Proprietary standards, 105–106  
Prototypes. *See also* scenarios; simulations.  
  architectural validation, 197–198, 204–206  
  pilots, 483  
  walkthroughs, 129–130  
Providers and responsibilities, 344, 345  
Proxies for stakeholders, 117–118  
  
**Q**  
Qualitative models, 161–162  
Quality properties, 13–16  
Quality triangle, 20–21  
Quantitative models, 162–163  
  
**R**  
Race conditions, 289–290  
Recovery. *See* disaster recovery.  
Reentrancy concerns, 274–275  
Relationships, modeling, 252





- Release process
  - ad hoc management, 470–471
  - automating, 467
  - defining, 302
- Replication, 227, 413–414
- Requirements
  - analysis, 57, 58–59, 75–76
  - architectural, limits and constraints, 98–99
  - architecture definition
    - analysis, 75–76
    - clarifying, 74
    - revisiting, 80–85
  - availability, 432–433
  - clarifying, 74
  - definition, architect's role in, 509
  - finding missing, 123
  - functional, 41, 98–99
  - hardware, 308–309, 314–315, 403
  - inappropriate, 447
  - network, 309–310
  - nonfunctional, 41
  - overlooked, 448–449
  - performance, 404–406
  - problems and pitfalls, 391–392, 447–449
  - response time, 405–406
  - reviewing, 412
  - revisiting, 80–85
  - routine maintenance, 340–341
  - routine procedures, 339
  - scalability, 406
  - scenarios from, 123–124
  - software concerns, 309
  - throughput, 406
- Resources, security
  - definition, 366
  - sensitive, identifying, 370–372
- Resources, system
  - allocation problems, 423–424
  - classes, 373
  - contention, Concurrency viewpoint, 288
  - sharing, 281, 417–418
- Response time, 398–401, 405–406
- Responsibilities, functional structure model, 232–233
- Responsiveness, concerns, 400
- Restore concerns, 330–332
- Reviews, architectural validation, 194–195, 204–206
- Risk assessment, 378–379
- Risks, identifying, 411
- Rolling back updates, 248
- Routine maintenance requirements, 340–341
- Routine procedure requirements, 339
- Runtime dependency analysis, 319
- Runtime platform models
  - activities, 314–315
  - client nodes, 311
  - constraints assessment, 315
  - deployment environment, designing, 314
  - element-to-node mapping, 312
  - hardware requirements, 314–315
  - mapping components to hardware, 314
  - network links, 312
  - notation, 313–314
  - offline storage hardware, 312
  - online storage hardware, 312
  - overview, 311
  - processing nodes, 311
  - UML deployment diagram, 313–314
- S**
- SAAM. *See* Software Architecture Analysis Method.
- Scalability. *See* Performance and Scalability perspective.
- Scenarios. *See also* prototypes; simulations.
  - applying, 128–130
  - architectural validation, 195–197, 199–204
  - in architecture definitions, 80–85
  - capturing, 125–128
  - communicating with stakeholders, 123

Scenarios, *continued*

- definition, 121
- driving the test process, 123
- effective use of, 131–132
- evaluating architecture, 122–123
- for failures, 132
- finding missing requirements, 123
- functional, 122
- identifying, 123–124
- identifying focused scenario set, 131
- input to architecture definition, 122
- live testing, 130
- from personal experience, 124
- prioritizing, 123–124
- prototyping, 129–130
- from requirements, 123–124
- simulations, 130
- sources for, 123–124
- stakeholder involvement, 132
- from stakeholders, 123–124
- system quality, 122, 132
- types of, 122
- uses for, 122–123
- using distinct scenarios, 131
- using scenarios early in the lifecycle, 131
- walkthroughs, 129

Sketch, architectural, 223

Scope

- architectural limits and constraints, 92, 93–94
- defining, 78
- definition, characteristics of, 94
- example, 91–92

Scope definition, architect's role in, 509

Screen magnifier, 477

Screen reader, 477

Secure failure, 381

Security

- administration, 386–387
- as afterthought, 392
- definition, 366
- embedded in code, 392
- implementation, designing, 376–378

- policies, 372–373, 387–388
- principals, 366
- principles, 379–382
- resources, 366
- technologies, integrating, 386

Self-describing data, 465

Sensitive operations, 373

Separation, security, 380

Service levels, defining, 432–433

Service types, identifying, 432

Shared resources, 281, 417–418

Shutdown concerns, system, 273

Simulations, 130. *See also* prototypes; scenarios.

Single point of failure, 446–447

Skeleton systems

- architectural validation, 198–199, 205–206
- creating, 80
- in the architecture definition process, 77, 80.

Sketches, 164

Software

- availability solutions, 444
- commonly used, 299
- configuration management, 466
- lifecycle, 86–89, 204–206
- patterns. *See* patterns.
- product development project, 116
- requirement concerns, 309
- variation points, 464–466

Software architects. *See* architects.

Software architecture. *See* architecture.

Software Architecture Analysis Method (SAAM), 195, 199.

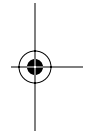
Solution architects, 63

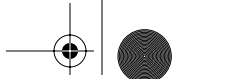
Source code structure, 302

Stakeholder-centric validation, 203–204

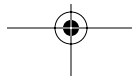
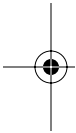
Stakeholders

- acquirers, 111, 112
- assessors, 111, 113
- capturing needs, 56–57
- classes of, 111–115
- communicating via scenarios, 123





- communication through views and viewpoints, 33
- communicators, 111, 113
- concerns
  - architectural principle, 73
  - Concurrency viewpoint, 275
  - definition, 20–21
  - Deployment viewpoint, 310–311
  - Development viewpoint, 295–296
  - Information viewpoint, 251
  - Operational viewpoint, 332
- criteria for, 110–111
- definition, 19
- developers, 111, 113
- engaging, 79
- entities as, 19–20
- example, 1–5
- groups as, 19–20
- identification, architect's role in, 508
- identifying for models, 166–167
- importance of, 21–22
- interests of, 20–21
- introduction, 1–6
- maintainers, 111, 113
- needs, functional structure model, 236
- people as, 19–20
- proxies, 117–118
- responsibilities, 118
- scenario involvement, 123–124, 132
- selecting, 109–111
- suppliers, 111, 113–114
- support staff, 111, 114
- system administrators, 111, 114
- testers, 111, 114
- users, 111, 115
- Standard extension points, 466
- Standards, architectural limits and constraints, 105–106
- Startup concerns, system, 273
- State entities, 282
- State models
  - action entities, 283
  - activities, 287
  - description, 282–283
  - designing state transitions, 287
  - entities, 282–283
  - event entities, 283
  - guards, 283
  - hierarchical states, 283
  - identifying states, 287
  - notation, 283–287
  - state entities, 282
  - transition entities, 283
- Statecharts, 257–258
- States
  - hierarchical, 283
  - identifying, 287
  - management concerns, 272–273
  - transitions, 257, 287
- Static structures, 12, 15–16, 252–254
- Strategies
  - applying perspectives to views, 52
  - architectural description (AD)
    - clarity, 179–180
    - conciseness, 179
    - currency, 180
    - glossaries, 182
    - precision, 180–181
  - architectural descriptions, sufficiency, 178
  - architectural limits and constraints, 106
  - architectural significance, 60
  - architecture definition, 86
  - constraints, 92
  - describing complex systems, 30
  - developing views, 32
  - good *versus* perfect, 86
  - managing stakeholders, 112
  - models, 160, 161
  - role of architects, 61
  - scope, 92
  - selecting modeling language, 162
  - selecting stakeholders, 110
  - stakeholder groups, 112, 118
  - stakeholder proxies, 118
  - view content, 31
- Structural decomposition, 254



**Styles**

- and architectural description (AD), 144–145
  - Asynchronous Data Replication, 149
  - benefits of, 142–144
  - Client/Server, 146
  - definition, 138
  - Distribution Tree, 149–150
  - identifying, 80–85
  - Integration Hub, 150–151
  - Layered Implementation, 147–148
  - Peer-to-Peer, 147
  - Pipes and Filters, 145–146
  - Publisher/Subscriber, 148–149
  - Tiered Computing, 146–147
  - Tuple Space, 151
  - types of, 145–151
- Suppliers, 111, 113–114
- Support concerns, 330
- Support models
- activities, 344–345
  - classes of incidents, 344, 345
  - escalation process, 344, 345
  - example, 345–348
  - groups needing support, 344
  - notation, 344
  - overview, 344
  - providers and responsibilities, 344, 345
- Support staff, 111, 114
- Synchronization concerns, 273
- System administrators, 111, 114
- System construction, architect's role in, 510
- System management tools, 350.
- System quality scenarios, 122, 132
- System structures, 12–13
- System testing, architect's role in, 510
- System-level concurrency models
- activities, 280–282
  - contention, 282
  - data-sharing mechanisms, 276
  - deadlocks, 281
  - description, 275

- execution coordination mechanisms, 276
- interprocess communication, 276–277
- lightweight processes, 276
- mapping elements to tasks, 280
- notation, 277–280
- procedure call mechanisms, 276
- process groups, 275–276
- processes, 275
- processes, prioritizing, 281
- shared resources, 281
- threading design, 280–281
- threads, 276
- threads, prioritizing, 281

**T**

- Tactics, *versus* design patterns, 43
- Tasks
- failure concerns, 274
  - mapping elements to, 272, 280
  - structure concerns, 271–272
- Technological neutrality, 74
- Technology
- assessment, security, 378
  - availability, improving, 441
  - compatibility concerns, 309
  - incompatibilities, 449
  - integration, security, 378
  - specialists *versus* architects, 65–66
- Technology dependency models
- activities, 319
  - notation, 318–319
  - overview, 317–318
  - runtime dependency analysis, 319
- Template views. *See* viewpoints.
- Test process, driving with scenarios, 123
- Test standardization concerns, 295
- Testers, 111, 114
- Testing. *See also* scenarios.
- automating, 467
  - live, 130
  - performance, 410–411
- Thin-client approach, definition, 15
- Threading design, 280–281



- Threads, 276, 281
  - Threat model, 374
  - Threats
    - concerns, 368
    - identifying, 373–376
    - mitigating, 378
  - Three-Peaks model, 75–76
  - Three-tier approach. *See* thin-client approach.
  - Throughput, 401–402, 406
  - Time sources, 390
  - Timeliness concerns, 245–246
  - Traceability, functional structure model, 231
  - Transaction logging, 443–444
  - Transaction management and recovery
    - concerns, 247–248
  - Trust and permissions, 259
  - Trust model, 372
  - Trusting external entities, 381
  - Turnaround time, concerns, 400
  - Twin-Peaks model, 75
  - Two-tier approach. *See* client/server approach.
- U**
- Undoing updates, 248
  - Unified modeling language (UML), 165
  - Upgrade concerns, 326
  - User acceptance testing, architect's role in, 510
  - User interface
    - Accessibility perspective, 474–478
    - handicapped users, 474–478
    - International perspective, 485–488
    - internationalization, 485–488
    - Usability perspective, 499–503
  - Users, 111, 115
  - Utilities, custom, 341
- V**
- Validation
    - with models, 170
    - reasons for, 192–193
  - recording results of, 206–207
  - in software lifecycle, 204–206
  - techniques
    - architecture-centric activities, 200–203
    - ATAM, 199–200
    - formal reviews, 194–195, 204–206
    - presentations, 193–194, 204–206
    - proof-of-concept systems, 197–198
    - prototypes, 197–198, 204–206
    - SAAM, 199
    - scenarios, 195–197, 199–204
    - skeleton systems, 198–199, 205–206
    - stakeholder-centric activities, 203–204
    - walkthroughs, 194–195, 204–206
  - Viewpoints. *See also specific viewpoints.*
    - benefits of, 32–34
    - catalogs, 35–36, 211–213
    - definition, 6, 31
    - groupings, 35–36
    - introduction, 1–6
    - managing complexity, 33
    - objectives, 31–32
    - pitfalls, 34–35
    - separating concerns, 32–33
    - stakeholder communication, 33
  - Views
    - benefits of, 32–34
    - definition, 6, 30
    - development focus, 34
    - fragmentation problems, 34–35
    - inconsistencies between, 34
    - introduction, 1–6
    - managing complexity, 33
    - pitfalls, 34–35
    - required content, 30–31
    - selecting wrong set of, 34
    - separating concerns, 32–33
    - stakeholder communication, 33
  - Views, consistency across
    - Concurrency views and
      - Deployment views, 360
      - Development views, 359–360





Views, consistency across, *continued*  
  Deployment views and, Operational views, 360  
  Functional views and  
    Concurrency views, 356  
    Deployment views, 357  
    Development views, 356  
    Information views, 355  
    Operational views, 357  
  Information views and  
    Concurrency views, 358  
    Deployment views, 358–359  
    Development views, 358  
    Operational views, 359  
  relationships, 354–355

Voice recognition, 477  
Volumetric models, 262  
Volumetrics, 266–267

**W**

Walkthroughs, 129, 194–195, 204–206.  
  *See also* scenarios.  
Waterfall development approach, 87–88  
Workload  
  characterization, 409  
  consolidating, 416  
Wolf, Alexander, 30.

