

INDEX

“4 + 1” view of architecture, 297–298

A

abstraction levels, 96–99
 acceptance tests. *See* testing.
 accuracy, requirements for, 261
 active storyboards, 134
 activity diagrams, 284–285
 actors
 business modeling, 62–63
 case study, 80–81, 245, 412
 definition, 53
 problem analysis
 case study, 80–81
 definition, 53
 identifying, 54
 use cases
 case study, 161–163
 definition, 150
 identifying and describing, 153
 use-case relationships, 155
 agendas, requirements workshops, 114
 agile requirements methods, 392–394, 400–403. *See also* risk mitigation methods.
 Allie, as product champion, 199
 alternative flow. *See* flow of events.
 Alyssa
 HOLIS team role, 38, 408
 interviewing, 106–107
 as product manager, 186, 412
 requirements workshops, 126–130, 415–418

 scope management, 221, 426
 vision document, 420
 ambiguity
 disambiguation, 274–277
 versus specificity, 271–277
 Analyst’s Summary, 106
 architecture, software, 296–299
 artifact sets, 358–359
 artifacts, definition, 29
 assessing development effort. *See* scope management.
 attributes, system, 232
 attributes of features, 99–100
 audit trails, 352–353
 availability, requirements for, 261

B

baseline for requirements
 case study, 426–427
 change management, 342–343
 CMM (Capability Maturity Model), 474–475
 establishing, 211–212
 Betty, requirements workshops, 127–130, 415–418
 Bill, partitioning subsystems, 77
 black-box testing, 305–307, 317–318
 brainstorming
 benefits of, 119–120
 cumulative voting, 124–125
 defining features, 123–124
 grouping ideas, 123

- brainstorming, *continued*
- hundred-dollar test, 124–125
 - idea generation, 120–122
 - idea reduction, 122–125
 - importance categorization, 125
 - live, 120–122
 - phases of, 120
 - prioritizing ideas, 124–125
 - pruning ideas, 122
 - Web-based, 126
- branding products, 197
- budget. *See* cost.
- budget information, in requirements, 235
- bugs. *See* defects; requirements, errors.
- building the right system. *See also* change management; test cases; testing; tests; traceability.
- “4 + 1” view of architecture, 297–298
 - case study, 433–435
 - design assurance, 317
 - design verification, 302–303
 - mapping requirements to design and code, 292–299
 - modeling software systems, 295–296
 - object orientation, 293–294
 - orthogonality problem, 292–293
 - process, 402
 - quality assessment
 - checklists, 361–368
 - in iterative development, 357–358
 - process, 360
 - quality, definition, 356
 - requirements artifacts sets, 358–359
 - software projects, 356–357
 - software architecture, 296–299
 - use cases
 - deriving test cases from, 306–307, 309–316
 - as requirements, 294–295
 - role in architecture, 298–299
 - scenarios, 309–312
- business modeling. *See also* problem analysis; systems engineering.
- actors, 62–63
 - business object model, 62
 - business use-case model, 62
 - entity description, 63–64
 - functional description, 62–63
 - publications, 62
 - purpose, 60
 - techniques, 60–64
 - translating to systems modeling, 64–65
 - UML (Unified Modeling Language), 61–64
 - use cases, 62–63
 - when to use, 65
- buyers. *See* customers.
- C**
- Capability Maturity Model (CMM). *See* CMM (Capability Maturity Model).
- capturing change requests, 344–346
- case study
 - actors, 80–81, 245, 412
 - background, 37–38, 407–408
 - baselining requirements, 426–427
 - building the right system, 433–435
 - characters in. *See specific names.*
 - constraints on the solution, 85, 413
 - interviewing, 106–107
 - pre- and post-conditions, 428
 - problem analysis, 78–79, 409–413
 - problem statement, 409–413
 - requirements organization, 170–171, 419
 - requirements workshops, 126–130, 415–418
 - scope management, 218–222, 424–427
 - software development team, 38–39, 408
 - solution system boundaries, 80–81
 - subsystems, 83–85
 - supplementary specification, 429–432
 - system definition, initial, 419–424
 - system definition, refining, 427–432
 - systems engineering, 81–83

- test cases, 433–435
- use cases
 - building, 158–163
 - flow of events, 247–249, 428
 - list of, 418
 - naming, 246–247
 - refining, 245–251
 - users and stakeholders
 - identifying, 80–81
 - needs assessment, 78, 412, 414–418
 - vision document, 419–424
- CCBs (change control boards), 345–346
- certificates, requirements workshops, 116
- champions, 184, 198–200
- change control. *See* change management.
- change control boards (CCBs), 345–346
- change control systems, 344–346
- change histories, 352–353
- change management. *See also* configuration management; scope management; traceability.
 - audit trails, 352–353
 - baselining requirements, 342–343
 - capturing change requests, 344–346
 - CCBs (change control boards), 345–346
 - change control systems, 344–346
 - change histories, 352–353
 - CMM (Capability Maturity Model), 477–478
 - configuration management, 349–350, 353
 - Easter Eggs, 341
 - hidden code, 341
 - hierarchical management, 347–349
 - illicit code, 341
 - planning for change, 342
 - prioritizing bug fixes, 344–346
 - process, 341–349, 402
 - programmer's role, 347–349
 - reasons for change, 337–340
 - request process, flow diagram, 346
 - requirements leakage, 340–341
 - restricting the change process, 343–344
 - ripple effects, 347–349
 - RUP (Rational Unified Process), 468–470
 - tools for, 351
 - unofficial changes, 340–341
- changing. *See* refining.
- CMM (Capability Maturity Model)
 - baselining requirements, 474–475
 - change management, 477–478
 - maturity levels, 474
 - requirements management, 473–478
 - traceability, 477
- collaborations, use cases, 299–302
- collecting requirements. *See* requirements, eliciting.
- commercial terms, defining, 194–195
- competitive advantage, creating, 196
- compiling interview results, 106–107
- complex systems, decomposing. *See* subsystems.
- concept testing, 193
- configuration management, 349–350, 353.
 - See also* change management.
- constraints
 - solution system, case study, 85, 413
 - system design, 237, 241
- construction phase, 28
- context-free interview questions, 102, 104–105
- contracting subsystems between teams, 75
- core message platforms, 196–197
- cost
 - allocation, hardware *versus* software, 73–74
 - fix exceeds cost of defect, 48
 - requirements errors
 - percentage of project budget, 13
 - by project stage, 10–12
 - repair costs, 12–13
 - risk mitigation methods, 386
 - software development, 6–7, 10–13
- cumulative voting, 124–125

customers
 involvement in scope management, 223–227
 negotiating with, 224–225
 product perception management, 195–197

D

decision process. *See* product managers;
 requirements, workshops.
decision tables/trees, 283–284
defects. *See also* requirements, errors.
 fix exceeds cost of defect, 48
 leakage through lifecycle phases, 12
 maximum rate, requirements, 261
 prioritizing fixes, 344–346
 tracking, 261
defining the system. *See* system definition.
Delta Vision documents, 177–181
demonstrating products, 197–198
dependency relationships, tracing, 321
derived requirements, 71–73
design assurance, 317. *See also* building the
 right system; testing; tests.
design constraints
 anecdote, 266
 definition, 241, 263
 handling, 265
 in requirements, 237
 sources of, 263–265
 versus true requirements, 265
design information, in requirements, 235
design verification, 302–303. *See also* building
 the right system; testing; tests.
design *versus* requirements, 238–239
developers
 generation gap, 73–74
 personality types, 33–34
 role in change management, 347–349
 users and stakeholders communication
 gap, 92
developing software. *See* software
 development; software lifecycle.

disambiguation, 274–277
disciplines, 29–30
documentation, as risk mitigation method,
 387–388

E

Earl, HOLIS team role, 38, 408
Easter Eggs, 341
E.C., requirements workshops, 127–130,
 415–418
elaboration phase, 28
eliciting requirements. *See* requirements,
 eliciting.
Elmer, requirements workshops, 127–130,
 415–418
Emily
 HOLIS team role, 38, 408
 requirements workshops, 127–130,
 415–418
 scope management, 219
emphasis technique for disambiguation,
 276–277
entities, describing, 63–64
ERDs (entity-relationship diagrams),
 285–286
errors. *See* defects; requirements, errors.
estimating development effort. *See* scope
 management.
extending use cases, 251–254
Extreme Programming (XP), 388–392

F

facilitators, choosing, 112, 114
failure, root causes, 7–13
features, product or system
 attributes, 99–100
 definition, 20, 97
 levels of abstraction, 96–99
 optimal number of, 98
 versus requirements, 233–234
scoping, 98–99

- stakeholder needs, 96–99
 - tracing to supplementary requirements, 327
 - tracing to use cases, 325–327
 - finite state machines, 281–283
 - flow of events
 - basic and alternate paths, 155–156
 - case study, 247–249, 428
 - definition, 151
 - specification template, 450–451
 - flowcharts, 284–285
 - “4 + 1” view of architecture, 297–298
 - functional *versus* nonfunctional software, 240–241
 - functions, describing, 62–63
 - functions, system, 232
- G**
- gathering requirements. *See* requirements, eliciting.
 - Gavin
 - HOLIS team role, 38, 408
 - refining the system definition, 427
 - Gene, HOLIS team role, 38, 408
 - generation gap, developers, 73–74
 - grouping ideas, 123
- H**
- hardware *versus* software
 - cost allocation, 73–74
 - requirements organization, 166–168
 - systems engineering, 72–74
 - hidden code, 341
 - hierarchical change management, 347–349
 - HOLIS project. *See* case study.
 - hundred-dollar test, 124–125
- I**
- idea generation, 120–122
 - idea reduction, 122–125. *See also* brainstorming.
- illicit code, 341
 - importance categorization, 125
 - inception phase, 28
 - including use cases in use cases, 254–255
 - inputs/outputs, system, 232–233
 - integration with RUP (Rational Unified Process), 470–471
 - interactive storyboards, 134–135
 - interface requirements, 71–73, 76
 - interviewing. *See also* requirements, eliciting; users and stakeholders, needs assessment.
 - Analyst’s Summary, 106
 - case study, 106–107
 - compiling results, 106–107
 - conducting the interview, 103–105
 - context-free questions, 102, 104–105
 - versus* questionnaires, 107–108
 - requirements repository, 106
 - sample questions, 104–105
 - solutions-context questions, 102–103
 - ISO 9000:2000, 478–482
 - isolating functions, 76
 - iterations, 28–29
 - iterative development
 - description, 27–31
 - quality assessment, 357–358
 - requirements and design, 239
- J**
- Jason, scope management, 219
 - Jenny, case study actor, 149
- K**
- keyword technique for disambiguation, 276–277
 - Kruchten, Philippe, 173
- L**
- labeling products, 197
 - levels of abstraction, 96–99

lifecycle. *See* software lifecycle.
light box exercise, 271–274
linkages. *See* traceability.
Louise, HOLIS team role, 38, 408
Lucy, requirements workshops, 127–130, 415–418

M

Magaziner, Elemer, 44
managing scope. *See* scope management.
mapping requirements to design and code, 292–299

Marcy
HOLIS team role, 38, 408
requirements workshops, 126–130, 415–418

Mark
HOLIS team role, 38, 408
refining the system definition, 427

marketing
branding, 197
collateral, 198
concept testing, 193
creating a competitive advantage, 196
defining commercial terms, 194–195
defining the user experience, 193–194, 197
end user materials, 193–194, 197
labeling, 197
managing buyer perceptions, 195–197
messaging, 195–197
product demos, 197–198
product positioning, 195–197
sales and marketing collateral, 198

Mary Had a Little Lamb, 274–276
maturity levels, CMM, 474
maximum defect rate, requirements, 261
mean time between failures (MTBF), 261
mean time to repair (MTTR), 261
memo sample, requirements workshops, 113–114
memorization heuristic for disambiguation, 276

message platforms, 196
messaging, 195–197
misunderstanding requirements. *See* ambiguity.
modeling systems. *See* business modeling; storyboarding; use cases.
modifying. *See* refining.
MTBF (mean time between failures), 261
MTTR (mean time to repair), 261

N

naming use cases, 154, 246–247
needs, users and stakeholders
assessing. *See also* interviewing; problem analysis; requirements, eliciting.
case study, 78, 412, 414–418
defining the user experience, 193–194, 197
description, 51–52
developing end user materials, 193–194, 197
level of abstraction, 96–99
quality checklist, 363
RUP (Rational Unified Process), 464–465
system features, 96–99
definition, 96
requirements management, 20
tracing to features, 324–325

nonfunctional software requirements
accuracy, 261
availability, 261
bug tracking, 261
versus functional, 240–241
maximum defect rate, 261
MTBF (mean time between failures), 261
MTTR (mean time to repair), 261
performance, 262
reliability, 261–262
supportability, 262–263
usability, 259–260
use cases, 251

- O**
- object orientation, 293–294
 - orthogonality problem, 292–293
- P**
- partitioning functions, 76, 77
 - passive storyboards, 134
 - performance, requirements for, 262. *See also* nonfunctional software requirements.
 - philosophy of requirements management, 376
 - positioning products, 195–197
 - pre- and post-conditions
 - case study, 428
 - definition, 151–152
 - refining, 249–250
 - refining use cases, 156
 - specification template, 451
 - principles of systems engineering, 69
 - priorities
 - bug fixes, 344–346
 - ideas, 124–125
 - scope management, 212
 - problem analysis. *See also* business modeling;
systems engineering; users and stakeholders.
 - actors
 - case study, 80–81
 - definition, 53
 - identifying, 54
 - agreeing on the problem, 45–46
 - case study, 78–79, 409–413
 - constraints, solution system
 - case study, 85
 - definition, 55
 - identifying, 55
 - sources of, 56
 - definition, 44
 - fishbone diagrams, 47
 - goal, 45
 - process, 400–401
 - quality checklist, 361–362
 - root cause analysis, 46–50
 - RUP (Rational Unified Process), 463–464
 - solution system boundaries
 - case study, 80–81
 - defining, 52–55
 - problem domain, 19–20
 - problem statements
 - case study, 409–413
 - description, 45–46
 - examples, 50, 79–80
 - problems, definition, 44
 - process integration with RUP (Rational Unified Process), 470–471
 - product managers
 - activities
 - branding, 197
 - concept testing, 193
 - creating a competitive advantage, 196
 - defining commercial terms, 194–195
 - defining the user experience, 193–194, 197
 - driving the vision, 187–189
 - end user materials, 193–194, 197
 - labeling, 197
 - managing buyer perceptions, 195–197
 - messaging, 195–197
 - product demos, 197–198
 - product planning, 189–192
 - product positioning, 195–197
 - sales and marketing collateral, 198
 - sponsoring use cases, 192–193
 - decision making, 185–187
 - managing software products, 185–187
 - product champions, 184, 198–200
 - product positioning, 195–197
 - products
 - branding, 197
 - champions, 184, 198–200
 - demos, 197–198
 - families, requirements organization, 168–170
 - labeling, 197

- products, *continued*
 planning, 189–192
 positioning, 195–197
 project-related information, in requirements,
 235
 projects. *See* software development.
 prototyping. *See* storyboarding.
 pruning ideas, 122
 pseudocode, 280–281
 publications
 business modeling, 62
- Q**
- quality, definition, 356
 quality assessment
 checklists, 361–368
 in iterative development, 357–358
 process, 360
 quality, definition, 356
 requirements artifacts sets, 358–359
 software projects, 356–357
 questionnaires *versus* interviews, 107–108.
See also interviewing.
- R**
- Raquel
 requirements workshops, 127–130,
 415–418
 scope management, 219
 Rational Unified Process (RUP). *See* RUP
 (Rational Unified Process).
 realizations, use case
 collaborations, 299–302
 role in testing, 317
 tracing, 328–329
 reducing complex systems. *See* subsystems.
 reducing project scope, 210–211, 216–218, 224
 refining system definition. *See* system
 definition, refining.
 refining use cases
 case study, 245–251
 description, 156
 evolutionary process, 243–244
 nonfunctional software requirements, 251
 pre- and post-conditions, 249–250
 scope definition, 244
 special requirements, 250–251
 reliability, requirements for, 261–262. *See also*
 nonfunctional software requirements.
 requirements. *See also* software lifecycle; use
 cases.
 artifacts sets, 358–359
 baseline
 case study, 426–427
 change management, 342–343
 CMM (Capability Maturity Model),
 474–475
 establishing, 211–212
 budget information, 235
 collecting. *See* requirements, eliciting.
 definition, 15
 describing system behavior, 232–233
versus design, 238–239
 design constraints, 237, 241
 design information, 235
 excluded topics, 235–237
versus features, 233–234
 flowdown, 71
 functional *versus* nonfunctional software,
 240–241
 in the iterative approach, 30–31
 iterative design, 239
 leakage, 340–341
 mapping to design and code, 292–299
 misunderstanding. *See* ambiguity.
 nonfunctional software
 accuracy, 261
 availability, 261
 bug tracking, 261
versus functional, 240–241
 maximum defect rate, 261
 MTBF (mean time between failures),
 261
 MTTR (mean time to repair), 261

- performance, 262
- reliability, 261–262
- supportability, 262–263
- usability, 259–260
- use cases, 251
- project-related information, 235
- quality assessment, 358–359
- sources of, 266
- specification methods
 - activity diagrams, 284–285
 - decision tables/trees, 283–284
 - ERDs (entity-relationship diagrams), 285–286
 - finite state machines, 281–283
 - flowcharts, 284–285
 - pseudocode, 280–281
 - state transition diagrams, 282
- system attributes, 232
- system elements, 232–233
- system functions, 232
- system inputs/outputs, 232–233
- systems engineering
 - derived, 71–73
 - interface, 71–73, 76
 - subsystem, 71–73
- test procedures, 235
- tracing. *See* traceability.
- versus* use cases, 233–234
- what *versus* how, 234–237
- requirements,
 - errors. *See also* defects.
 - cost, 10–13
 - frequency of, 9–10
 - organization
 - case study, 170–171, 419
 - defining subsystems, 166–168
 - future requirements, 170
 - hardware/software systems, 166–168
 - product families, 168–170
 - reducing complex systems, 166–168
 - requirements sets, 165–166
 - requirements specifications, 165–166
 - requirements, eliciting. *See also* needs;
requirements, workshops.
 - barriers to, 90–92
 - interviewing
 - Analyst’s Summary, 106
 - case study, 106–107
 - compiling results, 106–107
 - conducting the interview, 103–105
 - context-free questions, 102, 104–105
 - versus* questionnaires, 107–108
 - requirements repository, 106
 - sample questions, 104–105
 - solutions-context questions, 102–103
 - knowing what you don’t know, 91
 - product or system features
 - attributes, 99–100
 - definition, 97
 - levels of abstraction, 96–99
 - optimal number of, 98
 - scoping, 98–99
 - users and stakeholders, needs, 96–99
 - storyboarding
 - active boards, 134
 - anecdote, 139–141
 - interactive boards, 134–135
 - keeping it simple, 137–138, 139–141
 - passive boards, 134
 - purpose of, 135–136
 - tools for, 136–137
 - types of, 134–135
 - “Undiscovered Ruins” syndrome, 137–138
 - “Yes, But” syndrome, 137–138
 - “Undiscovered Ruins” syndrome, 91, 137–138
 - “User and Developer” syndrome, 92
 - user/developer communication gap, 92
 - “Yes, But” syndrome, 90–91, 137–138
 - requirements, management
 - choosing an approach, 399–403
 - CMM (Capability Maturity Model), 473–478

- requirements, management, *continued*
- concepts, 16–17
 - definition, 16
 - features, definition, 20
 - ISO 9000:2000, 478–482
 - philosophy of, 376
 - problem domain, 19–20
 - process, 403
 - purpose of, 474
 - RUP (Rational Unified Process), 461–471
 - software applications, 17–18
 - software development, 18
 - software requirements, definition, 21
 - solution domain, 20
 - techniques, 17–18
 - users and stakeholders, needs, 20
- requirements, workshops. *See also* needs;
- requirements, eliciting.
 - accelerating decisions, 109–110
 - agendas, 114
 - benefits of, 110
 - brainstorming
 - benefits of, 119–120
 - cumulative voting, 124–125
 - defining features, 123–124
 - grouping ideas, 123
 - hundred-dollar test, 124–125
 - idea generation, 120–122
 - idea reduction, 122–125
 - importance categorization, 125
 - live, 120–122
 - phases of, 120
 - prioritizing ideas, 124–125
 - pruning ideas, 122
 - Web-based, 126
 - case study, 126–130, 415–418
 - ensuring stakeholder participation, 111
 - facilitators, choosing, 112, 114
 - follow-up, 115, 117
 - logistics, 111
 - potential problems, 115, 117
 - preparing for, 110–114
 - running, 115–117
 - sample memo, 113–114
 - selling the concept, 110
 - tickets, 116
 - tricks of the trade, 115, 117
 - warm-up materials, 111–112
- requirements repository, 106
- requirements sets, 165–166
- resources, and scope management, 208
- Rick
- HOLIS team role, 38, 408
 - interviewing, 106–107
 - requirements workshops, 126–130, 415–418
 - scope management, 221, 426
- ripple effects of changes, 347–349
- risk assessment, and scope management, 214–216
- risk mitigation methods
- agility, 392–394
 - cost factors, 386
 - criticality of risk, 387
 - design and evaluation principles, 384–387
 - design goals, 384–387
 - documentation, 387–388
 - exchanging information, 386
 - matching to team size, 386
 - reasons for, 383–384
 - requirements techniques for, 385
 - robustness, 394–396
 - XP (Extreme Programming), 388–392
- robust risk mitigation methods, 394–396
- RUP (Rational Unified Process)
- change management, 468–470
 - description, 459–460
 - integration, 470–471
 - problem analysis, 463–464
 - process integration, 470–471
 - requirements management, 461–471
 - scope management, 466–468
 - structure of, 460–461
 - system definition, initial, 465–466

- system definition, refining, 468
 - users and stakeholders needs assessment, 464–465
 - Russ, HOLIS team role, 38, 408
 - Rusty, requirements workshops, 127–130, 415–418
- S**
- sales and marketing collateral, 198
 - scenarios, 309–312. *See also* business modeling; storyboarding; use cases.
 - scheduling projects, 189–192
 - scope management. *See also* change management.
 - anecdote, 209
 - assessing effort, 212–214
 - assessing risk, 214–216
 - baseline for requirements
 - case study, 426–427
 - change management, 342–343
 - CMM (Capability Maturity Model), 474–475
 - establishing, 211–212
 - managing changes, 225–227
 - case study, 218–222, 424–427
 - customer involvement, 223–227
 - description, 207–210
 - iterative approach, 30–31
 - negotiating with customers, 224–225
 - overextending initial estimates, 210
 - process, 402
 - quality checklist, 365
 - reducing initial estimates, 210–211, 216–218, 224
 - refining use cases, 244
 - resources, 208
 - RUP (Rational Unified Process), 466–468
 - setting priorities, 212
 - time, 208–210
 - underpromise and overdeliver, 225
 - waterfall model, 25
 - scoping features, 98–99
 - SEI-CMM model
 - baselining requirements, 474–475
 - change management, 477–478
 - maturity levels, 474
 - requirements management, 473–478
 - traceability, 477
 - SEI (Software Engineering Institute), 473. *See also* CMM (Capability Maturity Model).
 - software architecture, 296–299
 - software developers. *See* developers; teams.
 - software development
 - based on incomplete requirements, 39, 42
 - cost, 6–7, 10–13
 - goal of, 5–6
 - project-related information, in requirements, 235
 - requirements management, 18
 - root causes of success/failure, 7–13
 - scheduling, 189–192
 - techniques for business modeling, 60–64
 - software engineering. *See* systems engineering.
 - Software Engineering Institute (SEI), 473. *See also* CMM (Capability Maturity Model).
 - software lifecycle
 - construction phase, 28
 - disciplines, 29–30
 - elaboration phase, 28
 - inception phase, 28
 - iterations, 28–29
 - iterative approach, 27–31
 - lifecycle phases, 28
 - spiral model, 26–27
 - traditional approach, 23–27
 - transition phase, 28
 - waterfall model, 24–25
 - software projects. *See* software development.
 - software requirements. *See* requirements.
 - software *versus* hardware
 - cost allocation, 73–74
 - requirements organization, 166–168
 - systems engineering, 72–74

- solution domain, 20
- solution system boundaries, case study, 80–81
- solutions-context interview questions, 102–103
- spiral model, 26–27
- stakeholders. *See* users and stakeholders.
- state transition diagrams, 282
- storyboarding. *See also* requirements, eliciting.
 - active boards, 134
 - anecdote, 139–141
 - interactive boards, 134–135
 - keeping it simple, 137–138, 139–141
 - passive boards, 134
 - purpose of, 135–136
 - tools for, 136–137
 - types of, 134–135
 - “Undiscovered Ruins” syndrome, 137–138
 - use cases, 156–158
 - “Yes, But” syndrome, 137–138
- stovepipe problem, 75, 76
- subsystems
 - case study, 83–85
 - from complex systems, 69–71
 - contracting between teams, 75
 - partitioning and isolating function, 76, 77
 - requirements, 71–73
 - requirements information, organizing, 166–168
- success, common factors, 7–13
- supplemental specifications. *See also* system definition, refining.
 - case study, 429–432
 - design constraints
 - anecdote, 266
 - definition, 263
 - handling, 265
 - sources of, 263–265
 - versus* true requirements, 265
 - functional requirements, 258
 - linking to use cases, 266–267
 - nonfunctional requirements
 - accuracy, 261
 - availability, 261
 - bug tracking, 261
 - maximum defect rate, 261
 - MTBF (mean time between failures), 261
 - MTTR (mean time to repair), 261
 - performance, 262
 - reliability, 261–262
 - supportability, 262–263
 - usability, 259–260
 - requirements sources, 266
 - role of, 257
 - template, 267–268, 453–458
 - tracing to implementation, 329–330
 - tracing to test cases, 332–333
- supportability, requirements for, 262–263
- surveys. *See* interviewing.
- “sweet spot,” finding, 271–274
- system attributes, 232
- system behavior, describing. *See* system definition; use cases.
- system definition, initial. *See also* product managers; use cases.
 - case study, 419–424
 - process, 401
 - quality checklist, 364
 - requirements organization
 - case study, 170–171
 - defining subsystems, 166–168
 - future requirements, 170
 - hardware/software systems, 166–168
 - product families, 168–170
 - reducing complex systems, 166–168
 - requirements sets, 165–166
 - requirements specifications, 165–166
 - RUP (Rational Unified Process), 465–466
 - vision documents
 - case study, 419–424
 - components, 174
 - Delta Vision versions, 177–181
 - importance of, 173

- in legacy systems, 180–181
 - in product lifecycle, 177–181
 - template, 175–176, 437–447
 - tracking product evolution, 177–181
 - system definition, refining. *See also*
 - supplemental specifications; use cases, refining.
 - ambiguity *versus* specificity, 271–277
 - attributes, 232
 - case study, 427–432
 - describing system behavior, 232–233
 - disambiguation, 274–277
 - finding the “sweet spot,” 271–274
 - functions, 232
 - inputs/outputs, 232–233
 - process, 402
 - quality checklist, 365–367
 - RUP (Rational Unified Process), 468
 - system elements, 232–233
 - technical methods
 - activity diagrams, 284–285
 - decision tables/trees, 283–284
 - ERDs (entity-relationship diagrams), 285–286
 - finite state machines, 281–283
 - flowcharts, 284–285
 - pseudocode, 280–281
 - state transition diagrams, 282
 - system elements, 232–233
 - system functions, 232
 - system inputs/outputs, 232–233
 - systems engineering. *See also* business modeling; problem analysis.
 - case study, 81–83
 - complex systems, decomposing. *See* subsystems.
 - cost allocation, hardware *versus* software, 73–74
 - definition, 68
 - hardware *versus* software, 72–74
 - principles of, 69
 - as problem analysis technique, 68
 - recommendations, 76–77
 - requirements
 - derived, 71–73
 - flowdown, 71
 - interface, 71–73, 76
 - subsystem, 71–73
 - stovepipe problem, 75, 76
 - subsystems
 - case study, 83–85
 - from complex systems, 69–71
 - contracting between teams, 75
 - partitioning and isolating function, 76, 77
 - requirements, 71–73
 - systems engineers. *See* developers.
- T**
- team skills
 - balance and coverage, 36
 - overview, 35–36
 - Skill 1. *See* problem analysis.
 - Skill 2. *See* users and stakeholders, needs assessment.
 - Skill 3. *See* system definition, initial.
 - Skill 4. *See* scope management.
 - Skill 5. *See* system definition, refining.
 - Skill 6. *See* building the right system.
 - team wide, 35–36
 - teams
 - assembling, case study, 38–39, 408
 - case study, 408
 - contracting subsystems between, 75
 - history of, 34–35
 - organization, 36–37
 - organization chart, 38
 - size, and risk mitigation methods, 386
 - templates
 - flow of events specification, 450–451
 - memo, requirements workshops, 113–114
 - pre- and post-condition specification, 451
 - sample interview questions, 104–105
 - supplemental specifications, 267–268, 453–458

- templates, *continued*
 - use-case specification, 449–452
 - vision document, 175–176, 437–447
 - test cases
 - case study, 433–435
 - data values, 314–316
 - definition, 307
 - deriving from use cases, 306–307, 309–316
 - role of, 309
 - tracing from supplemental specifications, 332–333
 - tracing from use cases, 330–332
 - testing
 - black box, definition, 305–307
 - black box, *versus* white box, 317–318
 - relationship of artifacts, 308
 - tracing requirements, 330–333
 - tests
 - conditions, 313–314
 - coverage, 307, 316
 - items, 307
 - plans, 307
 - procedures, 235, 307
 - results, 307
 - scripts, 307
 - tickets, requirements workshops, 116
 - time, and scope management, 208–210
 - tools for
 - change management, 351
 - storyboarding, 136–137
 - traceability, 333–334
 - traceability
 - change management, 348–349, 351–352
 - CMM (Capability Maturity Model), 477
 - database analysis, 334–335
 - definition, 320
 - dependency relationships, 321
 - features to supplementary requirements, 327
 - features to use cases, 325–327
 - hierarchy, 322–323
 - realizations to implementation, 329
 - requirement to requirement, 323–327
 - requirements to implementation, 327–330
 - requirements to testing, 330–333
 - role in system development, 319–320
 - spreadsheet analysis, 334–335
 - supplementary requirements to
 - implementation, 329–330
 - supplementary requirements to test cases, 332–333
 - system definition domain, 323–327
 - tools for, 333–334
 - traceability matrix, 324
 - upstream *versus* downstream, 321
 - use cases to realizations, 328–329
 - use cases to test cases, 330–332
 - user needs to features, 324–325
 - traceability relationships, 320–322
 - Tracy, HOLIS team role, 38, 408
 - traditional approach, 23–27
 - transition phase, 28
 - translating business models to systems
 - models, 64–65
- U**
- UML activity diagrams, 284–285
 - UML (Unified Modeling Language), 61–64
 - underpromise and overdeliver, 225
 - understanding the problem. *See* problem analysis; requirements; users and stakeholders, needs assessment.
 - “Undiscovered Ruins” syndrome, 91, 137–138
 - Unified Modeling Language (UML), 61–64
 - updating. *See* refining.
 - usability, requirements for, 259–260
 - use-case-driven design, 299–300
 - use cases. *See also* business modeling; problem analysis; storyboarding.
 - actors
 - case study, 161–163, 245
 - definition, 150
 - identifying and describing, 153
 - use-case relationships, 155

- basic and alternate paths. *See* flow of events.
- benefits of, 148
- building, 152–156
- case study, 158–163, 418
- collaborations, 299–302
- definition, 149
- deriving test cases from, 306–307, 309–316
- in the design model, 299–302
- elements of, 149–152
- extending, 251–254
- flow of events
 - basic and alternate paths, 155–156
 - case study, 247–249, 428
 - definition, 151
 - specification template, 450–451
- including in other use cases, 254–255
- linking to supplemental specifications, 266–267
- modeling software systems, 62–63, 295–296
- naming
 - case study, 246–247
 - importance of, 154
- pre- and post-conditions
 - case study, 428
 - definition, 151–152
 - refining, 249–250
 - refining use cases, 156
 - specification template, 451
- realizations
 - collaborations, 299–302
 - role in testing, 317
 - tracing, 328–329
- refining
 - case study, 245–251
 - description, 156
 - evolutionary process, 243–244
 - nonfunctional software requirements, 251
 - pre- and post-conditions, 156, 249–250
 - scope definition, 244
 - special requirements, 250–251
- as requirements, 294–295
- versus* requirements, 233–234
- role in architecture, 298–299
- role of product manager, 192–193
- scenarios, 309–312
- specification template, 449–452
- storyboarding, 156–158
- tracing to test cases, 330–332
- user interface design, 156–158
- “User and Developer” syndrome, 92
- user experience
 - defining, 193–194, 197
 - end user materials, 193–194, 197
- user interface design, 156–158
- users and stakeholders. *See also* features,
 - product or system; interviewing.
 - developer communication gap, 92
 - identifying
 - case study, 80–81
 - description, 50–52
 - needs
 - definition, 96
 - requirements management, 20
 - tracing to features, 324–325
 - needs assessment. *See also* problem analysis; requirements, eliciting.
 - case study, 78, 412, 414–418
 - defining the user experience, 193–194, 197
 - description, 51–52
 - developing end user materials, 193–194, 197
 - level of abstraction, 96–99
 - quality checklist, 363
 - RUP (Rational Unified Process), 464–465
 - system features, 96–99
 - requirements workshop participation, 111
 - stakeholder definition, 50
- User’s Bill of Rights, 260

V

verifying the design, 302–303. *See also*
building the right system; testing.
version control. *See* change management.
vision documents
case study, 419–424
components, 174
Delta Vision versions, 177–181
importance of, 173
in legacy systems, 180–181
in product lifecycle, 177–181
template, 175–176, 437–447
tracking product evolution, 177–181

W

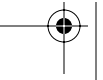
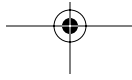
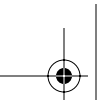
waterfall model, 24–25
white box testing *versus* black box, 317–318


X

XP (Extreme Programming), 388–392

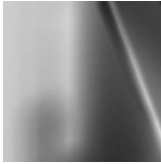
Y

“Yes, But” syndrome, 90–91, 137–138




The logo for InformIT, with 'inform' in a bold, lowercase sans-serif font and 'IT' in a larger, all-caps sans-serif font.The website address www.informit.com in a white sans-serif font on a black background.The main title of the guide in a bold, black, all-caps sans-serif font.The section title for the articles section in a white sans-serif font on a black background.A close-up photograph of several black keyboard keys, including the '0' and '1' keys.

Keep your edge with thousands of free articles, in-depth features, interviews, and IT reference recommendations – all written by experts you know and trust.

The section title for the online books section in a white sans-serif font on a black background.A blurred image of a book cover, showing the spine and part of the front cover.

Answers in an instant from **InformIT Online Book's** 600+ fully searchable on line books. For a limited time, you can get your first 14 days **free**.

The logo for Safari Tech Books Online, featuring the word 'Safari' in a large, bold, serif font, with 'POWERED BY' in a smaller font above it and 'TECH BOOKS ONLINE' in a smaller font below it.The section title for the catalog section in a white sans-serif font on a black background.A photograph of a stack of several books, showing the spines and the edges of the pages.

Review online sample chapters, author biographies and customer rankings and choose exactly the right book from a selection of over 5,000 titles.

Wouldn't it be great
if the world's leading technical
publishers joined forces to deliver
their best tech books in a common
digital reference platform?

They have. Introducing
InformIT Online Books
powered by Safari.

■ **Specific answers to specific questions.**

InformIT Online Books' powerful search engine gives you relevance-ranked results in a matter of seconds.

■ **Immediate results.**

With InformIT Online Books, you can select the book you want and view the chapter or section you need immediately.

■ **Cut, paste and annotate.**

Paste code to save time and eliminate typographical errors. Make notes on the material you find useful and choose whether or not to share them with your work group.

■ **Customized for your enterprise.**

Customize a library for you, your department or your entire organization. You only pay for what you need.

Get your first 14 days FREE!

For a limited time, InformIT Online Books is offering its members a 10 book subscription risk-free for 14 days. Visit <http://www.informit.com/onlinebooks> for details.

POWERED BY
Safari
TECH BOOKS ONLINE

informIT
Online Books

informit.com/onlinebooks



Register Your Book

at www.awprofessional.com/register

You may be eligible to receive:

- Advance notice of forthcoming editions of the book
- Related book recommendations
- Chapter excerpts and supplements of forthcoming titles
- Information about special contests and promotions throughout the year
- Notices and reminders about author appearances, tradeshows, and online chats with special guests



Contact us

If you are interested in writing a book or reviewing manuscripts prior to publication, please write to us at:

Editorial Department
Addison-Wesley Professional
75 Arlington Street, Suite 300
Boston, MA 02116 USA
Email: AWPro@aw.com

Visit us on the Web: <http://www.awprofessional.com>

