

INDEX

A

Acceptance testing

- automated, 161–162
- description, 160–162
- FIT approach, 161–162
- just-in-time elaboration, 234
- principles, 156

Accountability

- paradigm shifts, 84
- teams, 113

Agile. *See also* Software agility.

- adoption trends, 10–11
- customer challenges and misconceptions
 - announcing release content, 284–285
 - multiple annual releases, 285–286
 - press releases, 286
- evangelists, case study, 257
- manifesto, 9–10
- release train
 - end-to-end use cases, 243–244
 - interdependencies, 246–247
 - lessons learned, 241
 - measuring progress, 244–245
 - principles, 241–242
 - release management team, 244
 - synchronization, 242–243
 - system-level patterns, 245–246
 - themes, 243–244
 - vision, 243–244
- at scale. *See* Scaling agile.
- Agile, methods
 - introduction, 6–7
 - list of, 8

most widely used, 8

paradigm shifts

- accountability, 84
- architecture, 78
- coding practices, 78–79
- design processes, 78
- dividing big jobs into little ones, 84
- estimation, 84
- implementation practices, 78–79
- incremental code development, 83–85
- management culture, 77
- measures of success, 76–77
- ownership, 84
- planning, 80
- quality assurance, 79–80
- requirements gathering, 78
- risk detection, 85
- scheduling, 80
- scope versus schedules, 80–81
- testing, 79–80
- time boxes, 81–83
- tracking, 84

Agile and Iterative Development: A Manager's Guide, 19

Agile Project Management, 110

Agile/Scrum Master

- case study, 252–253
- team leader, 106

Agile UP (Agile Unified Process), 15, 60–61

Agility, BSC (balanced scorecard), 321–322

Allen, Thomas, 262

Ambler, Scott, 15

Architects, 108, 253

- Architectural debt, 185–186
- Architectural runway
 - building, 204–211
 - extending
 - lean, pull-based approach, 211
 - skills required, 208
 - synchronizing with the iteration, 209–211
 - impediments to scaling agile, 89
- Architectural spikes, 38–39
- Architecture
 - component-based systems, 203–204
 - critical objectives, 205
 - data view, 201
 - definition, 195–197
 - deployment view, 201
 - enterprise class systems, 203–204
 - FDD (Feature-Driven Development), 199–200
 - fragile nature of, 207–208
 - implementation view, 201
 - logical view, 201
 - needs assessment, 202–203
 - paradigm shifts, 78
 - process view, 201
 - refactoring, 201–202, 208–209
 - RUP (Rational Unified Process), 200–201
 - Scrum, 198–199
 - systems of scale, 201–202
 - temporal nature of, 207–208
 - use-case view, 201
 - XP (Extreme Programming), 197–198
- Architecture-driven development, 56
- Articles. *See* Books and publications.
- Automated build management, 173–174
- B**
- Ba concept, 293
- Baby steps, XP principle, 35
- Backlogs. *See* Product backlog.
- Balanced scorecard (BSC). *See* BSC (balanced scorecard).
- Barriers. *See* Impediments.
- Baseline executables, 54
- Basic Unified Process, 60
- Beck, Kent, 13, 29, 31
- Big Up-Front Design (BUFD), 30
- BMC Software case study, 255–261
- Boehm, Barry, 19–20, 196
- Booch, Grady, 52–53, 200
- Books and publications
 - Agile and Iterative Development: A Manager's Guide*, 19
 - Agile Project Management*, 110
 - “Distributed Agile Development and the Death of Distance,” 262
 - Good to Great*, 110
 - Java Modeling in Color with UML*, 70
 - Lean Software Development: An Agile Toolkit . . .*, 65
 - Managing the Development of Large Software Systems*, 17
 - A Practical Guide to Feature Driven Development*, 70
 - “The Toyota Way: 14 Management Principles . . .,” 302–303
 - The Unified Software Development Process*, 52
- Boundary conditions, 112
- Brainstorming, 228–230
- Brooks, Frederick, 19–20
- Brownbags, 307
- BSC (balanced scorecard)
 - aggregation, 324
 - agility, 321–322
 - converting to alphabetic grade, 323–324
 - efficiency, 319–320
 - implementing, 322–324
 - metrics at scale, 322–324
 - overview, 319
 - quality, 320–321
 - quantifying matrix elements, 322–323
 - value delivery, 321
- BUFD (Big Up-Front Design), 30

- Build verification tests (BVTs), 174–175
- Built-in instability, 43
- Burn-down charts, 134–135
- Business benefits of software agility, 11–13
- Business owners, large-scale releases, 152
- Business performance, measuring
 - agility measures, 311–312
 - BSC (balanced scorecard)
 - aggregation, 324
 - agility, 321–322
 - converting to alphabetic grade, 323–324
 - efficiency, 319–320
 - implementing, 322–324
 - metrics at scale, 322–324
 - overview, 319
 - quality, 320–321
 - quantifying matrix elements, 322–323
 - value delivery, 321
 - metrics
 - iteration, 313
 - process, 313–317
 - “process police,” 318
 - project, 312–313
 - release, 313
 - at scale, 322–324
 - team self-assessment, 318
 - “process police,” 318
 - radar charts, 317
 - scaling
 - agility, 321–322
 - efficiency, 319–320
 - overview, 319
 - quality, 320–321
 - value delivery, 321
 - team performance, 312–317
 - team self-assessment, 318
- BVTs (build verification tests), 174–175
- C**
- Cadence calendar, 135–137
- Cards, stories, 231
- Case studies. *See* Distributed development, case studies.
- Centralized repositories, 254
- Certified Scrum Masters (CSM), 42
- Change management organizations. *See* Organizational change.
 - projects, 21–22, 54
- Change verification, RUP, 55
- Chats, 307
- “Chicken and pig” metaphor, 127
- CIO-led seminars, 307
- Coad, Peter, 70
- Coarse-grained plans, 115
- Coding practices, paradigm shifts, 78–79
- Collaboration, teams, 112–113
- Collins, Jim, 110
- Collocation of teams, 35–36, 89
- Communication
 - conference rooms, 264
 - e-mail, 264
 - infrastructure, 263
 - instant messaging, 264
 - on-site visits, 262–263
 - phone connectivity, 263–265
 - presence center, 264
 - shuttle advocacy, 262–263
 - speaker phones, 264
 - teams, 112–113
 - whiteboards, 265
 - wikis, 264–265
 - XP core value, 33
- Competitive advantage
 - characteristics of winners, 5
 - effects of agile, 272–273
 - software development methods, 6
- Component architecture, RUP, 55
- Component testing, 156, 162
- Component-based systems, 54, 203–204
- Conference rooms, 264
- Configuration management, DSDM, 69
- Confirmation, stories, 231
- Conformance to release dates, 185

- Construction iterations, 58–59
 - Construction lifecycle, 57
 - Continuous integration
 - automated build management, 173–174
 - benefits of, 171
 - BVTs (build verification tests), 174–175
 - case study, 257–258
 - definition, 169
 - overview, 170–171
 - process description, 172–175
 - source code integration, 172–173
 - successful builds, 175–177
 - XP key practice, 37
 - Continuous risk attack, 53
 - Conversations, stories, 231
 - Core values of XP, 33–34
 - Corporate culture, impediments to scaling
 - agile, 92
 - Cost of error correction, 31–33
 - Courage, XP core value, 34
 - CSM (Certified Scrum Masters), 42
 - Cultural environment, impediments to scaling agile, 90
 - Cunningham, Ward, 161
 - Customers
 - agile challenges and misconceptions
 - announcing release content, 284–285
 - multiple annual releases, 285–286
 - press releases, 286
 - in the agile framework, 117
 - meeting needs of, 260
 - planning, 117
 - RUP, value to, 54
 - in teams, 88–89
 - Customers, effects of agile
 - competitive advantage, 272–273
 - marketing, 272–275, 280
 - product management, 273–275
 - releases
 - chasing agile, 278–279
 - development, 279–284
 - external, 279–284
 - ignoring agile, 277–278
 - size and frequency, 275–276
 - sales, 272–273, 280
- D**
- Daily stand-up meetings
 - define/build/test component team, 112–113
 - guidelines for, 133–134
 - iteration tracking, 133–134
 - organizational change, 302
 - tracking iterations, 133–134
 - Data view, architecture, 201
 - Define/build/test component team. *See also* Teams.
 - accountability, 113
 - Agile/Scrum Master team leader, 106
 - architects, 108
 - boundary conditions, 112
 - case study, 251–254, 255–261
 - choosing the right people, 110–111
 - collaboration, 112–113
 - communication, 112–113
 - creating, 109–113
 - daily stand-up meetings, 112–113
 - developers, 107
 - distributed teams, 114
 - eliminating functional barriers, 113
 - eliminating impediments, 106
 - enforcing rules, 106
 - facilitating, 106
 - functional silos, 104–105
 - large-scale releases, 150
 - leadership versus management, 111–112
 - lifecycle of simple story, 102–104
 - product owner, 107
 - purpose of the mission, 112
 - quality assurance, 108–109
 - roles and responsibilities, 106–109
 - schedule of the mission, 112
 - scope of the mission, 112
 - self-managing, 109–113
 - self-organizing, 109–113

- team leader, 106
 - testers, 107–108
 - understanding the mission, 112
 - vision of the mission, 112
 - Defined process control, 45–46
 - DeLuca, Jeff, 70
 - Deployment view, architecture, 201
 - Design processes, paradigm shifts, 78
 - Developer testing. *See* Unit testing.
 - Developers, as team members, 107
 - Development manager, case study, 252–253
 - “Dial tone” network bandwidth, 268
 - Discipline axis, 57–58
 - Distributed development
 - case studies
 - accelerating successful adoption, 258
 - agile evangelists, 257
 - Agile/Scrum Master, 252–253
 - architects, 253
 - BMC Software, 255–261
 - centralized repositories, 254
 - continuous integration, 257–258
 - define/build/test component team, 251–254, 255–261
 - development manager, 252–253
 - expanding agile, 261
 - follow-on activities, 260–261
 - high distribution, 255–261
 - identifying impediments, 258
 - incorporating engineering practices, 261
 - low distribution, 251–254
 - meeting customer needs, 260
 - Ping Identity Corporation, 251–254
 - product owner, 253
 - releasability, 260–261
 - reorganizing teams, 257
 - requirements analyst, 253
 - requirements architect, 258
 - Scrums, 254
 - shuttle advocacy, 254
 - SQA (Software Quality Assurance), 253
 - support for change, 258
 - team coordination, 258, 259
 - time-to-value acceleration, 259–260
 - communication
 - conference rooms, 264
 - e-mail, 264
 - infrastructure, 263
 - instant messaging, 264
 - on-site visits, 262–263
 - phone connectivity, 263–265
 - presence center, 264
 - shuttle advocacy, 262–263
 - speaker phones, 264
 - whiteboards, 265
 - wikis, 264–265
 - overview, 249–250
 - tooling the infrastructure
 - backlog management, 266
 - “dial tone” network bandwidth, 268
 - in early iterations, 268–269
 - early testing, 267
 - just-in-time requirements elaboration, 266–267
 - networking infrastructure, 268
 - overview, 265–266
 - planning iterations, 267
 - planning releases, 267
 - project reporting, 266
 - source code management, 267–268
 - tracking iterations, 267
 - tracking releases, 267
 - VPN (virtual private networking), 268
- Distributed teams, 114, 129
- Documented specifications, impediments to scaling agile, 90
- Documenting requirements, 215–217
- DoD (U.S. Department of Defense), 19–20
- Double inspection cycle, 46–48
- DSDM (Dynamic Systems Development Method). *See also* FDD (Feature-Driven Development); Lean Software Development.

DSDM, *continued*

- accessing, 70
- background, 66–67
- configuration management, 69
- core practices, 68–69
- core principles, 66
- definition, 65–66
- fundamental tenet, 67–68
- modeling, 69
- MoSCoW prioritization, 68
- philosophy, 66
- prototyping, 69
- Public Version, 67–68
- scope management, 67–68
- testing, 69
- time-boxing, 68
- Web site, 65

E

- Easel, 14
- Eclipse Foundation, 60
- Eclipse Process Framework, 60
- Efficiency, 319–320
- Elaboration of requirements
 - iterations, 58–59
 - just-in-time
 - acceptance test cases, 234
 - brainstorming, 228–230
 - feature realization, 228
 - incremental implementation, 230
 - LRM (last responsible moment), 229
 - optimizing requirements and design, 228–230
 - requirements, 230
 - use cases, 232–234
 - user stories, 230–232
 - lifecycle, 56
 - stories, 230–232
- E-mail, 264
- Empirical process control, 45–46
- Energized work, 36

- Engineering, training, 307
- Enterprise architect, 152
- Error correction, cost over time, 31–33
- Estimable stories, 232
- Estimation, paradigm shifts, 84
- Executing iterations
 - accepting the iteration, 131–132
 - basic process, 129–130
 - delivering the story, 131
 - development, 130–131
 - responsibility, 130
 - story completion, 131
- Executive management, organizational change, 299–304
- Executive sponsors, organizational change, 296–297, 301
- Executive-level managers, large-scale releases, 152
- Extreme Programming (XP). *See* XP (Extreme Programming).

F

- Facilitating, teams, 106
- FDD (Feature-Driven Development).
 - See also* DSDM (Dynamic Systems Development Method); Lean Software Development.
 - architecture, 199–200
 - best practices, 71–73
 - build schedules, 72–73
 - class, 72
 - code ownership, 72
 - configuration management, 73
 - definition, 70
 - developing by feature, 71–72
 - domain object modeling, 71
 - feature teams, 72
 - history of, 70
 - inspections, 72
 - reporting results, 73
 - visibility of results, 73

Feature points, 184
Feature realization, 228
Feature set, estimating, 143–144
Feature-level document, 216. *See also* Vision document.
Feedback, XP core value, 34
Feedback from pilot(s), 307
Fine-grained plans, 115
FIT (Framework for Integrated Tests), 161–162
Fixed functionality mandates, 92
Focus on executables, 54
Forrester Research, 10–11, 319
Fowler, Martin, 169
Functional silos, 104–105
Functional testing. *See* Acceptance testing.

G

Good to Great, 110

H

High degree of distribution, impediments to scaling agile, 93
Highsmith, Jim, 76, 110, 198
Humanity, XP principle, 34–35
Hurdles. *See* Impediments.

I

IBM, 60–61

Impediments to scaling agile
 case study, 258
 from the enterprise
 corporate culture, 92
 existing policies and procedures, 91
 fixed functionality mandates, 92
 fixed schedules, 92
 high degree of distribution, 93
 inter-team frictions, 93
 people organization, 93
 process management organizations, 91
 project management organizations, 91

 from the methods themselves
 architectural runway, 89
 collocation of teams, 89
 cultural environment, 90
 customers in teams, 88–89
 documented specifications, 90
 physical environment, 90
 requirements analysis, 90
 small team size, 88
 organizational, 186–187
 organizational change, 292–293, 298–300, 302
Implementation practices, paradigm shifts, 78–79
Implementation view, architecture, 201
Inception iterations, 58
Inception lifecycle, 56
Incremental code development, paradigm shifts, 83–85
Incremental design, 38
Independent stories, 231
Information radiators, 307
Informative workspace, 36
Inherently testable systems, 156
Inspection, Scrum, 46
Instant messaging, 264
Integration. *See* Continuous integration.
Intellectual property, nature of, 21
Intentional architecture. *See* Architecture.
Interdependencies, agile release train, 246–247
INVEST rule for stories, 231–232
Iteration plan, 128
Iterations
 adjusting, 132–135
 advantages of, 124
 anatomy of, 117–118
 assessment metrics, 181
 burn-down charts, 134–135
 cadence calendar, 135–137
 call for action, 183–184

Iterations, *continued*

- daily stand-up meetings, 133–134
- defining, 117
- definition, 123
- executing
 - accepting the iteration, 131–132
 - basic process, 129–130
 - delivering the story, 131
 - development, 130–131
 - responsibility, 130
 - story completion, 131
- functionality achievement, 182
- illustrations, 116, 121
- key process indicators, 182–183
- metrics, 313
- optimal length, 123–124
- organizational change, 303
- planning, 267
- planning meeting
 - development team responsibility, 126
 - distributed teams, 129
 - guidelines, 128–129
 - holding the meeting, 127–128
 - the iteration plan, 128
 - participants, 126–127
 - preparing for, 126
 - product owner responsibility, 126
 - responsibilities, 126
- process model, 125
- qualitative assessment, 183
- quality indicators, 182–183
- quantitative assessment, 180–183
- retrospective analysis, 180–184
- RUP tenet, 55–56
- story achievement, 182
- testing patterns, 164–167
- tracking, 132–135, 267
- two-week standard, 123–124
- unit testing, 158

Iterative development, RUP, 54

J

- Jacobsen, Ivar, 200
- Java Modeling in Color with UML*, 70
- Job satisfaction, 12
- Just-in-time requirements elaboration
 - acceptance test cases, 234
 - brainstorming, 228–230
 - feature realization, 228
 - incremental implementation, 230
 - LRM (last responsible moment), 229
 - optimizing requirements and design, 228–230
 - requirements, 230
 - tooling the infrastructure, 266–267
 - use cases, 232–234
 - user stories, 230–232

K

- Kroll, Per, 60
- Kruchten, Philippe, 200

L

- Lagging teams, 245
- Last responsible moment (LRM), 224, 229
- Leadership versus management, 111–112
- Lean Software Development. *See also* DSDM (Dynamic Systems Development Method); FDD (Feature-Driven Development).
 - agile principles, 65
 - cell-based manufacturing, 64
 - continuous process improvement, 64
 - cross-training, 64
 - cycle time reduction, 64–65
 - introduction, 63–64
 - inventory, owning and maintaining, 64
 - manufacturing cost reductions, 64–65
 - work-in-process reductions, 64
- Lean Software Development: An Agile Toolkit . . .*, 65
- Lifecycle iteration types, 58–59
- Load testing, 164

- Logical view, architecture, 201
LRM (last responsible moment), 224, 229
- M**
- Management
 Agile Project Management, 110
 culture, paradigm shifts, 77
 executive management, organizational change, 299–304
 versus leadership, 111–112
 “The Toyota Way: 14 Management Principles . . .,” 302–303
 Managing the Development of Large Software Systems, 17
Manifesto, 9–10
Marketing, effects of agile, 272–275, 280
Marketing requirements document (MRD).
 See Feature-level document.
Martens, Ryan, 75
Measures of success, paradigm shifts, 76–77
Measuring
 business performance. *See* Business performance, measuring.
 release progress, 244–245
 Scrum, organizational change, 308–309
 success, 76–77
Meetings
 conference rooms, 264
 daily stand-up, 112–113
 on-site visits, 262–263
 phone connectivity, 263–265
 presence center, 264
 release status review, 148–149
 shuttle advocacy, 262–263
 speaker phones, 264
 whiteboards, 265
Methods, software agility. *See* Agile, methods.
Metrics, business performance. *See also*
 Business performance, measuring.
 iteration, 313
 process, 313–317
 “process police,” 318
 project, 312–313
 release, 313
 at scale, 322–324
 team self-assessment, 318
MoSCoW prioritization, 68
MRD (marketing requirements document).
 See Feature-level document.
Multilearning, Scrum, 44
- N**
- Native scaling, 7
Negotiable stories, 232
Networking infrastructure, 268
- O**
- Object-oriented roots of RUP, 52–59
Objectory Process, 52–53
Obstacles. *See* Impediments.
On-site visits, 262–263
OpenUP (Open Unified Process), 15, 60
Optimizing requirements and design, 228–230
Organizational change
 assessment, 304–305
 ba concept, 293
 daily stand-ups, 302
 difficulties of, 297–298
 eliminating impediments, 292–293, 298–300, 302
 empirical versus planned processes, 291–292
 executive management, 299–304
 executive sponsors, 296–297, 301
 iterations, 303
 letting go, 294–295
 organizational expansion, 306–307
 overview, 289–290, 304
 pilot preparation, 304–305
 pilot project(s), 305–306
 preparation for, 295–298

Organizational change, *continued*

principles of, 290

releases, 303–304

requirements for, 290–295

Scrum

achieving impact, 307–308

adjusting, 308–309

brownbags, 307

chats, 307

CIO-led seminars, 307

feedback from pilot(s), 307

information radiators, 307

measuring, 308–309

organizational, 296

rolling out, 304–309

software process, 296

suggested readings, 307

war stories, 307

Scrum, training

engineering, 307

product development, 307

Scrum Master, 306–307

Scrum/Agility, 307

Scrum Masters, 296–297

software productivity, 298–299

Zen of Scrum, 292–293

Organizational transfer of learning, 44

Overlapping development phases, 44

Ownership, paradigm shifts, 84

P

Pair programming, 36, 38

Palmer, Stephen, 70

Papers. *See* Books and publications.

Paradigm shifts

accountability, 84

architecture, 78

coding practices, 78–79

design processes, 78

dividing big jobs into little ones, 84

estimation, 84

implementation practices, 78–79

incremental code development, 83–85

management culture, 77

measures of success, 76–77

ownership, 84

planning, 80

quality assurance, 79–80

requirements gathering, 78

risk detection, 85

scheduling, 80

scope versus schedules, 80–81

testing, 79–80

time boxes, 81–83

tracking, 84

Patterns, 164–167, 245–246

People organization, impediments to scaling
agile, 93

Performance testing

business. *See* Business performance,
measuring.

description, 162–164

principles, 156

Phone connectivity, 263–265

Physical environment, impediments to
scaling agile, 90

Pilot project(s), 304–307

Ping Identity Corporation case study, 251–254

Planning

coarse-grained plans, 115

customers, 117

fine-grained plans, 115

generalized framework, 116–120

iterations

anatomy of, 117–118

defining, 117

illustrations, 116, 121

tooling the infrastructure, 267

large-scale releases, 150–154

paradigm shifts, 80

product backlog, 117

release plan, 120

- releases, 144–147, 267
 - anatomy of, 119
 - defining, 118
 - features, 118
 - illustrations, 116, 121
 - planning, 119
 - requirements allocation, 119
 - Planning meeting
 - development team responsibility, 126
 - distributed teams, 129
 - guidelines, 128–129
 - holding the meeting, 127–128
 - the iteration plan, 128
 - participants, 126–127
 - preparing for, 126
 - product owner responsibility, 126
 - responsibilities, 126
 - Policies and procedures, impediments to scaling agile, 91
 - Poppendiek, Mary and Tom, 64, 127
 - A Practical Guide to Feature Driven Development*, 70
 - PRD (product's requirements document). *See* Feature-level document.
 - Presence center, 264
 - Procedures and policies, impediments to scaling agile, 91
 - Process management organizations, impediments to scaling agile, 91
 - Process metrics, 313–317
 - “Process police,” 318
 - Process view, architecture, 201
 - Product backlog
 - agile requirements, 220–221, 224–225
 - managing, 266
 - planning, 117
 - versus* refactoring, 208–209
 - in the release process, 117
 - Product development, training, 307
 - Product management, effects of agile, 273–275
 - Product owner
 - agile requirements, role of, 220
 - on the agile team, 107
 - conversion to requirements analyst, 253
 - definition, 107
 - large scale releases, 152
 - planning meeting, responsibility, 126
 - Scrum, role, 42
 - Productivity increases, 11–12
 - Product's requirements document (PRD). *See* Feature-level document.
 - Programmer testing. *See* Unit testing.
 - Project management organizations, impediments to scaling agile, 91
 - Project metrics, 312–313
 - Prototyping, DSDM, 69
 - Public Version, DSDM, 67–68
 - Publications. *See* Books and publications.
- Q**
- Quality
 - assurance, 79–80, 108–109
 - BSC (balanced scorecard), 320–321
 - improvement, 12
 - indicators, iterations, 182–183
 - integrating, 54
 - monitoring, 55
 - scaling, and business performance, 320–321
 - XP principle, 35
 - Quarterly cycles, 37
- R**
- Radar charts, 317
 - Rational Objectory Process, 53
 - Rational Software Corporation, 15
 - Rational Unified Process (RUP). *See* RUP (Rational Unified Process).
 - Recommended reading. *See* Books and publications.
 - Refactoring architecture, 201–202, 208–209

- Refactors, 185–186
- Reflection, XP principle, 35
- Releasability, case study, 260–261
- Release plan, 120, 146–147
- Release train. *See* Agile, release train.
- Releases
 - anatomy of, 119
 - architectural debt, 185–186
 - conformance to release dates, 185
 - customer challenges and misconceptions
 - announcing release content, 284–285
 - multiple annual releases, 285–286
 - press releases, 286
 - defining, 118, 141–144
 - effects of agile
 - chasing agile, 278–279
 - development, 279–284
 - external, 279–284
 - ignoring agile, 277–278
 - size and frequency, 275–276
 - estimating the feature set, 143–144
 - feature points, 184
 - features, 118
 - fixed periodic release dates, 142–143
 - illustrations, 116, 121
 - large scale
 - business owners, 152
 - component teams, 150
 - enterprise architect, 152
 - executive-level managers, 152
 - multiple teams, 153–154
 - organizational structure, 151
 - planning, 150–154
 - product owners, 152
 - Scrum of Scrums, 150–152
 - steering committee, 152–153
 - tracking, 154
 - metrics, 313
 - organizational change, 303–304
 - organizational impediments, 186–187
 - participants, 145
 - planning, 119, 144–147, 267
 - planning process, 145–146
 - planning responsibilities, 146
 - qualitative assessment, 186
 - quantitative assessment, 184–186
 - refactors, 185–186
 - requirements allocation, 119
 - retrospective analysis, 184–187
 - roadmaps, 149
 - schedule-driven, 141–142
 - scheduling, 141–144
 - small
 - benefits of, 139–141
 - business risk reduction, 140–141
 - illustration, 116
 - increased responsiveness, 139–140
 - sample scenario, 139–140
 - status review meeting, 148–149
 - testing patterns, 164–167
 - tracking, 147–149, 267
 - value delivery, 184–185
- Requirements. *See also* Stories.
 - analysis, impediments to scaling agile, 90
 - analysts, case study, 253
 - architects, case study, 258
 - decay, waterfall method assumptions, 25–26
 - definition decay, waterfall method
 - assumptions, 21
 - documenting, 215–217
 - feature-level document, 216
 - gathering, paradigm shifts, 78
 - just-in-time elaboration, 230
 - management, RUP, 54–55
 - for organizational change, 290–295
 - software requirements specification, 216–217
- Requirements, agile approach
 - distinguishing characteristics, 217–218
 - product backlog, 220–221
 - product owner, role of, 220
 - RUP (Rational Unified Process), 221–222
 - Scrum, 220–221

- supplemental specifications, 221, 222
- use-case model, 221, 222
- vision document, 221–222
- XP (Extreme Programming), 218–219
- Requirements, scalable agile approach
 - backlogs, 224–225
 - communicating requirements, 225–226
 - elaboration, just-in-time
 - acceptance test cases, 234
 - brainstorming, 228–230
 - feature realization, 228
 - incremental implementation, 230
 - LRM (last responsible moment), 229
 - optimizing requirements and design, 228–230
 - requirements, 230
 - use cases, 232–234
 - user stories, 230–232
 - need for a vision, 223–226
 - roadmaps, 226–228
 - vision document, 223–224
 - way-advanced data sheets, 224
- Requirements pyramid
 - overview, 213–214
 - software requirements, 215
 - solution features, 215
 - stakeholder needs, 214–215
 - traditional approaches, 215–217
- Respect, XP core value, 34
- Retrospective analysis, 180–187
- Risk detection, paradigm shifts, 85
- Risk reduction, small releases, 140–141
- Roadblocks. *See* Impediments.
- Roadmaps, 149, 226–228
- Royce, Walker, 17, 200
- Royce, Winston, 17
- Rumbaugh, Jim, 52–53
- RUP (Rational Unified Process)
 - agile requirements, 221–222
 - agile variants, 60–61
 - Agile UP (Agile Unified Process), 60–61
 - applicability, 61–62
 - architecture, 200–201
 - architecture-driven development, 56
 - baseline executables, 54
 - best practices, 54–55
 - change management, 54
 - change verification, 55
 - characteristics of, 51–52
 - component architecture, 55
 - component-based systems, 54
 - construction, lifecycle, 57
 - construction iterations, 58–59
 - continuous risk attack, 53
 - definition, 51
 - discipline axis, 57–58
 - elaboration, lifecycle, 56
 - elaboration iterations, 58–59
 - focus on executables, 54
 - fundamental tenet, 55–56
 - history of, 14
 - inception, lifecycle, 56
 - inception iterations, 58
 - integral quality, 54
 - iteration, 55–56
 - iterative development, 54
 - lifecycle iteration types, 58–59
 - light versions of, 15
 - monitoring quality, 55
 - object-oriented roots, 52–59
 - OpenUP (Open Unified Process), 60
 - principles, 53–54
 - process model, 57
 - requirements management, 54–55
 - roots of, 52–59
 - teamwork, 54
 - time axis, 56–57
 - transition, lifecycle, 57
 - transition iterations, 59
 - use case-centric development, 56
 - value to the customer, 54
 - visual modeling, 55
- RUP for Extreme Programming XP Plug-in*, 15

S

Sales, effects of agile, 272–273, 280

Scaling agile

architecture, 201–202

business performance

agility, 321–322

efficiency, 319–320

overview, 319

quality, 320–321

value delivery, 321

impediments, of the enterprise

corporate culture, 92

existing policies and procedures, 91

fixed functionality mandates, 92

fixed schedules, 92

high degree of distribution, 93

inter-team frictions, 93

people organization, 93

process management organizations, 91

project management organizations, 91

impediments, of the methods themselves

architectural runway, 89

collocation of teams, 89

cultural environment, 90

customers in teams, 88–89

documented specifications, 90

physical environment, 90

requirements analysis, 90

small team size, 88

large scale releases

business owners, 152

component teams, 150

enterprise architect, 152

executive-level managers, 152

multiple teams, 153–154

organizational structure, 151

planning, 150–154

product owners, 152

Scrum of Scrums, 150–152

steering committee, 152–153

tracking, 154

metrics, 322–324

native scaling, 7

team practices, 7–8, 325–326

Schedule-driven releases, 141–142

Schedules

agile component release, 238–242

agile release train

end-to-end use cases, 243–244

interdependencies, 246–247

lessons learned, 241

measuring progress, 244–245

principles, 241–242

release management team, 244

synchronization, 242–243

system-level patterns, 245–246

themes, 243–244

vision, 243–244

cadence calendars, 135–137

conformance to release dates, 185

fixed periodic release dates, 142–143

impediments to scaling agile, 92

iterations, 135–137

mission, 112

paradigm shifts, 80

releases, 141–144

versus scope, 80–81

system integration phase, 239

waterfall method, assumptions, 23–26

Schwaber, Ken

organizational change, 298

process metrics, 313

Scrum development, 13–14, 41

Scope

management, DSDM, 67–68

of the mission, 112

versus schedules, paradigm shifts, 80–81

triage, waterfall method assumptions,
24–25

Scrum

adaptation, 46

agile requirements, 220–221

- applicability, 48–49
 - architecture, 198–199
 - built-in instability, 43
 - case study, 254
 - defined process control, 45–46
 - definition, 41
 - double inspection cycle, 46–48
 - empirical process control, 45–46
 - fundamental tenet, 45–46
 - inspection, 46
 - key characteristics, 8
 - key practices, 14, 44–45
 - multilearning, 44
 - organizational change
 - achieving impact, 307–308
 - adjusting, 308–309
 - brownbags, 307
 - chats, 307
 - CIO-led seminars, 307
 - feedback from pilot(s), 307
 - information radiators, 307
 - measuring, 308–309
 - organizational, 296
 - rolling out, 304–309
 - software process, 296
 - suggested readings, 307
 - war stories, 307
 - organizational change, training
 - engineering, 307
 - product development, 307
 - Scrum Master, 306–307
 - Scrum/Agility, 307
 - organizational transfer of learning, 44
 - overlapping development phases, 44
 - philosophical roots, 42–43
 - principles of, 43–44
 - process model, 46–48
 - Product Owner role, 42
 - roles of participants, 42
 - scope of usage, 8
 - Scrum Master role, 42
 - self-organizing project teams, 43
 - sprints, 44–45
 - subtle control, 44
 - Team role, 42
 - visibility, 46
 - word origin, 42–43
 - Zen of Scrum, 292–293
- Scrum Alliance, 42
 - Scrum Masters
 - organizational change, 296–297
 - role, 42
 - training, 306–307
 - Scrum of Scrums, 150–152
 - Scrum/Agility, training, 307
 - Self-managing teams, 109–113
 - Self-organizing teams, 43, 109–113
 - Shine Technologies, 11
 - Sho, Fuijo, 303
 - Shuttle advocacy, 254, 262–263
 - Simplicity, XP core value, 34
 - Slack time, 37
 - Small releases
 - benefits of, 139–141
 - business risk reduction, 140–141
 - increased responsiveness, 139–140
 - sample scenario, 139–140
 - Small stories, 232
 - Software agility. *See also* Agile.
 - business benefits, 11–13
 - job satisfaction, 12
 - productivity increases, 11–12
 - quality improvement, 12
 - team morale, 12
 - time to market, 13
 - Software architecture. *See* Architecture.
 - Software productivity, organizational change, 298–299
 - Solution features, requirements pyramid, 215
 - Source code
 - integration, 172–173
 - management, 267–268

Speaker phones, 264
Specifications, impediments to scaling agile, 90
Sprints, 44–45
SQA (Software Quality Assurance), 253
SRS (software requirements specification), 216–217
Stakeholder needs, requirements pyramid, 214–215
Stand-up meetings. *See* Daily stand-up meetings.
Status review meeting, 148–149
Steering committee, large scale releases, 152–153
Stories. *See also* Requirements.
 aspects of, 231
 attributes of, 231–232
 building, 103
 cards, 231
 completion, 131
 confirmation, 231
 conversations, 231
 criteria, 219
 defining, 102–103
 delivering, 131
 elaboration, 230–232
 estimable, 232
 independent, 231
 INVEST rule, 231–232
 just-in-time elaboration, 230–232
 lifecycle of, 102–104
 negotiable, 232
 small, 232
 testable, 232
 testing, 103–104
 valuable, 232
 writing, 231–232
 XP key practice, 36–37
Story achievement, iterations, 182
Subtle control, 44
Successful builds, 175–177

Suggested reading. *See* Books and publications.
Supplemental specifications, 221, 222
Sutherland, Jeff, 13–14, 41
Synchronization, agile release train, 242–243
System design specification. *See* Software requirements specification.
System integration, waterfall method assumptions, 22–23
System integration phase, 239
System testing, 156, 162–164
Systems requirements specification. *See* Feature-level document.

T

TDD (Test-Driven Development), 159–160
Team role, Scrum, 42
Teams. *See also* Define/build/test component team.
 case study, 257–259
 collocation of, impediments to scaling agile, 89
 customers in, impediments to scaling agile, 88–89
 impediments to scaling agile, 93
 lagging, 245
 morale, 12
 multiple, large scale releases, 153–154
 performance, measuring, 312–317
 release management, 244
 RUP teamwork, 54
 scaling agile, 325–326
 self-assessment, 318
 self-organizing, 43
 size, impediments to scaling agile, 88
 skill sets required, 36
Technical specification. *See* Software requirements specification.
Ten-minute builds, 37
Testable stories, 232
Testers, 107–108

- Test-first development. *See* TDD (Test-Driven Development).
- Test-first programming, 37–38
- Testing
- acceptance
 - automated, 161–162
 - description, 160–162
 - FIT (Framework for Integrated Tests), 161–162
 - principles, 156
 - BVTs (build verification tests), 174–175
 - component, 156, 162
 - developer. *See* Unit testing.
 - DSDM, 69
 - functional. *See* Acceptance testing.
 - inherently testable systems, 156
 - introduction, 155–156
 - load, 164
 - paradigm shifts, 79–80
 - patterns, 164–167
 - performance, 156, 162–164
 - principles, 156–158
 - programmer. *See* Unit testing.
 - stories, 103–104
 - strategy, 165
 - system, 156, 162–164
 - unit
 - in the course of iteration, 158
 - description, 158–160
 - history of, 158
 - key quality forecasters, 182
 - principles, 156
 - and TDD, 159–160
 - white-box, 156
- “The Toyota Way: 14 Management Principles . . .,” 302–303
- Themes, 243–244
- Time axis, 56–57
- Time to market, 13
- Time-boxing, 68, 81–83
- Time-to-value acceleration, 259–260
- TogetherSoft, 70
- Tooling the infrastructure
- backlog management, 266
 - “dial tone” network bandwidth, 268
 - in early iterations, 268–269
 - early testing, 267
 - just-in-time requirements elaboration, 266–267
 - networking infrastructure, 268
 - overview, 265–266
 - planning iterations, 267
 - planning releases, 267
 - project reporting, 266
 - source code management, 267–268
 - tracking iterations, 267
 - tracking releases, 267
 - VPN (virtual private networking), 268
- Tracking
- coarse-grained plans, 115
 - customers, 117
 - fine-grained plans, 115
 - generalized framework, 116–120
 - iterations
 - adjusting, 132–135
 - anatomy of, 117–118
 - defining, 117
 - illustrations, 116, 121
 - planning, 267
 - tracking, 132–135, 267
 - large scale releases, 154
 - paradigm shifts, 84
 - product backlog, 117
 - release plan, 120
 - releases
 - anatomy of, 119
 - defining, 118
 - documentation, 148–149
 - features, 118
 - illustrations, 116, 121
 - outcomes, 148–149
 - planning, 119, 267

Tracking, releases, *continued*
 requirements allocation, 119
 status review meeting, 148
 tracking, 267

Training, 307

Transition iterations, 59

Transition lifecycle, 57

Turner, Richard, 196

Two-week standard iteration, 123–124

U

The Unified Software Development Process, 52

Unit testing
 in the course of iteration, 158
 description, 158–160
 history of, 158
 principles, 156
 and TDD, 159–160

U.S. Department of Defense (DoD), 19–20

Use case-centric development, 56

Use cases
 agile release train, 243–244
 converting to test cases, 234
 just-in-time elaboration, 232–234
 standard elements, 233

Use-case model, 221, 222

Use-case view, architecture, 201

V

Valuable stories, 232

Value delivery, 184–185, 321

Value to the customer, 54

Visibility, Scrum, 46

Vision
 agile release train, 243–244
 of the mission, 112
 need for, 223–226

Vision document, 221–224. *See also* Feature-level document.

Visual modeling, 55

VPN (virtual private networking), 268

W

War stories, 307

Waterfall method
 corrective agile methods, 26–27
 flowchart, 18
 history of, 17–19
 largest failure factor, 19
 problems with, 19–20
 underlying assumptions
 change management, 21–22
 changing business behavior, 21
 delivery schedules, 23–26
 nature of intellectual property, 21
 requirements decay, 25–26
 requirements definition, 21
 scope triage, 24–25
 system integration, 22–23
 “*Yes, But*” syndrome, 21

Way-advanced-data-sheets, 224

Weekly cycles, 37

Whiteboards, 265

White-box testing, 156

Whole teams, 36

Wikis, 264–265

Writing stories, 231–232

X

XP (Extreme Programming)
 agile requirements, 218–219
 applicability, 39–40
 architectural spikes, 38–39
 architecture, 197–198
 baby steps, 35
 basic principles, 34–35
 BUFD (Big Up-Front Design), 30
 characteristics of, 29
 collocation of teams, 35–36
 communication, 33
 continuous integration, 37
 controversial practices, 30
 core values, 33–34
 cost of error correction, 31–33

courage, 34
energized work, 36
extreme nature of, 30–31
feedback, 34
fundamental tenets, 31–33
graphical representation, 39
humanity, 34–35
incremental design, 38
informative workspace, 36
key practices, 13, 35–38
pair programming, 36, 38
process model, 38–39
quality, 35
quarterly cycles, 37

reflection, 35
respect, 34
scope of usage, 8
simplicity, 34
slack time, 37
stories, 36–37
ten-minute builds, 37
test-first programming,
 37–38
weekly cycles, 37
whole teams, 36

y

“Yes, But” syndrome, 21, 140