

# Index

## A

- Adapter pattern, 317
- adaptive planning, 210
  - agile principles and, 375
  - entrepreneurs and, 204
  - example, 214
  - organizational culture, 216
- adopting XP, 43–63
- Agile Estimating and Planning (Cohn), 255
- Agile methods, 9
  - principles, 353
  - successes and, 6
- agile thrashing, 24
- Alexander, Christopher, 381
- analysis paralysis, 327
- analysis phase, 60–61
  - adopting XP and, 60
  - eliminating, 273
  - XP lifecycle and, 17
- architects, 23, 27, 33
  - adopting XP and, 35
  - Alexander, Christopher, 381
  - process improvement (retrospective alternative) and, 97
  - simple design and, 243
- architecture, 20, 257
- asynchronous integration, 186
  - broken builds and, 186
  - efficiency of, 187
  - multistage builds and, 187
  - performance tests and, 336
- automation, 178, 282

## B

- baby steps, 365, 367
- background noise, 118
- batman, 39, 47, 241
- Beck, Kent, 44, 296
  - acknowledgment of, xvii
  - simplicity, on, 315
  - XP differences and, 24
- big ball of mud, 298
- big visible chart, 84
- binary chop, 171
- bit rot, 387
  - (see also technical debt)
- boundary testing, 343
- brainstorming, 92, 235
- branches (version control), 170, 174
- breakthroughs in continuous design, 322
- breeding grounds (bugs), eliminating, 162
- Brooks' Law, 108, 372
- bugs, 25 (see no bugs)
- build machine (integration machine), 52, 157
- burn-up charts, 145, 228
- business analysts, 28, 32
  - agile principles and, 361
  - customer proxies and, 46
  - poor communication, accommodating, 99
- business rules/logic (see domain knowledge)

## C

- cache coherency, 335
- calendar time, 100
- challenged projects, 4
- change, challenges of in teams, 53
- charters, xvii, 342

We'd like to hear your suggestions for improving our indexes. Send email to [index@oreilly.com](mailto:index@oreilly.com).

- charts, 84–85
- check in (version control), 169
- check out (version control), 169
- CI (continuous integration) servers, 185
- clarifying practices, 25
- Class, Responsibility, Collaborator (see CRC cards)
- coaches, 28, 34, 47
  - mentors and, 12, 35
  - staffing, 38
- Cockburn, Alistair, xviii, 104, 113, 196, 386
  - face-to-face communication, 273
  - stories and, 253
- cocktail party effect, 114
- coddling nulls, 304
- code
  - preexisting, 49 (see preexisting code)
  - refactoring, 50
  - self-documenting, 317
- code inspections, 77
- code smells, 303–304
- codebase, single, 173
- coding phase, 61
  - adopting XP and, 61
  - XP lifecycle and, 21
- coding standards, 133–138
  - adopting, 117
  - bugs and, 148
  - collective code ownership and, 170
  - creating, 39, 99, 133–135
  - disagreement and, 136
  - enforcing, 136
  - incremental design and, 269
  - legacy code and, 137
  - pair programming and, 57
  - programmers and, 33
  - starting out, 56
  - XP lifecycle and, 20
- cohesion, 385
- collaboration, 99–151, 203
  - adopting XP and, 54, 161
  - agile principles and, 314
  - benefits of, 114, 192
  - customers, importance to, 120–124
  - design and, 269
  - étude, 100
  - miracle of, 221
  - pair programming and, 69–79
  - planning game and, 195, 271
  - requirements and, 235
  - team size and, 38
  - vision and, 191
  - XP lifecycle and, 18
- collective code ownership, 20, 33, 153, 191–195
  - benefits of, 191–192
  - bugs and, 162
  - coding standards, 135
  - handoff documentation, alternative to, 196
  - pair programming, 77
  - planning sessions and, 239
  - programmers and, 33, 268
  - refactoring and, 312
  - ten-minute build and, 180
  - velocity, improving and, 268
  - version control and, 170, 175, 194
  - XP lifecycle and, 21
- colocation of teams (see sit together)
- command injection, 344
- commit (version control), 169
- commit build, 187
- commitments, 105, 237
  - bugs and, 163
  - defending, 229
  - design, 314, 318
  - “done done” and, 153, 159
  - iteration, 110, 237, 239, 244–252
  - last responsible moment and, 41
  - planning horizons and, 214–216
  - product managers, 30
  - release, 159, 224–232, 268
  - slack and, 246
  - trust and, 102–106, 110
  - velocity and, 42, 268
- communication, 113, 195
  - (see also collaboration)
  - agile principles and, 363
  - bug databases and, 166
  - customer tests and, 278–285, 284
  - design and, 315, 328
  - documentation and, 195–198
  - domain knowledge, 124
  - executive sponsor and, 140
  - informative workspace and, 83
  - iteration demo and, 138
  - iteration planning and, 233, 243
  - osmotic, 113, 118
  - pair programming and, 74, 114
  - planning game and, 222
  - refactoring and, 311

- releases and, 213, 228
- requirements, 28, 268, 273, 277, 284
- sitting together, importance of, 45, 112–114, 118
- stand-up meetings and, 131
- team size and, 39, 268
- ubiquitous language and, 125
- vision and, 201–206
  - XP lifecycle and, 18, 19
  - XP value, 354
- compensation (pay), 36, 44, 110, 150
- concepts (XP), 39–42
- concurrent model (version control), 170
- contingency, 227
- continuous design, 322
- continuous integration, 33, 62, 169, 183–191
  - adopting XP and, 56, 61, 186
  - asynchronous, 186–189
  - collective code ownership and, 193
  - “done done”, 157
  - frequent releases of software and, 208
  - informative workspace and, 83
  - legacy code and, 58
  - machine, 52, 184, 189
  - multistage, 187
  - performance tests and, 338
  - problems, 187
  - programmers and, 33
  - refactoring and, 311
  - release planning and, 208, 218
  - script, 184–185
  - servers, 185–186
  - slow builds and, 186
  - synchronous, 186
  - ten-minute build and, 180
  - version control, 171
  - XP lifecycle and, 20
- cooperative games, 219
- costs in planning game, 219
- coupling, 385
- CRC (Class, Responsibility, Collaborator) cards, 83, 327, 384
- cross-functional teams, 28
- CRUD (Create, Read, Update, Delete), 256, 344
- Cunningham, Ward, xvii, 75, 282
- custom development, 120
- customer huddles, 265
- customer review boards, 121
- customer reviews, 24, 274
- customer tests, 24, 25, 271, 278–285, 281
  - acceptance tests, contrasted with, 24, 283
  - bugs and, 161
  - communicating with, 278–282
  - customers and, 28, 31, 278
  - documentation as, 195, 278–282
  - domain experts and, 31
  - “done done” and, 156
  - ISO 9001 and, 371
  - iteration demo and, 139
  - no bugs, writing, 162
  - planning and, 214
  - programmers and, 33, 278
  - requirements and, 274
  - testers and, 34, 282
  - version control and, 171, 172
  - XP lifecycle and, 19
- customer-centricity, 255
- customers, 28–32
  - coaches, 202
  - domain experts and, 31
  - frequent releases and, 210
  - incremental requirements and, 273–278
  - on-site (see on-site customers)
  - programmer discord, 107
  - real (see real customers)
  - value, 253
- customers tests
  - iteration demos, 139

**D**

- daily deployment, 210, 218
- Daily Scrum, 129
- data classes, 304
- data clumps, 303
- data type attacks, 344
- databases
  - authority over, 44
  - bug/issue trackers, 58, 165
  - “done done” and, 156
  - incremental design and, 328
  - nulls and, 304
  - refactoring, 311
  - requirements trackers, 273
  - risks (historical), 224
  - ten-minute build and, 180
  - tests and, 299
  - version control and, 174
- deadlines, 4
  - Brooks’ Law and, 108

- customers and, 102
- energized work and, 82
- estimating and, 262, 269
- iteration, 138, 239, 261
- iteration demo and, 138
- release planning and, 209, 212, 269
- risk management and, 229, 231
- success and, 229
- timebox, 240
- velocity and, 261
- writing software and, 5
- death marches, 82, 103
- debugging
  - diff, 171
  - estimating, 256
  - reducing, 184, 367
  - reverting (version control) and, 170
  - spike solutions and, 333
  - stories, 256
  - ten-minute build and, 182
  - test-driven development and, 162, 285, 297, 301
- decoupling, 385
- defects, reporting, 146
  - (see also no bugs)
- delivering value, 5, 375–380
- demo, iteration (see iteration demo)
- deployment, 20
  - adopting XP and, 61
  - continuous integration and, 183
  - daily, 218
  - “done done” and, 156
  - feedback from, 121
  - iteration demo and, 141–143
  - phase, 19, 61
  - product manager and, 30
  - real customers and, 203
  - tagging (version control) and, 173
  - ten-minute build and, 180
  - weekly, 141–143, 156
  - XP lifecycle and, 18–27
- Describe, Demonstrate, Develop process, 278
- design, 18
  - continuous, 329
  - incremental (see incremental design)
  - phase, 18, 61
  - predictive, 303
  - quality, 383
  - reflective, 303
  - simple (see simple design)
  - skills, importance of, 49
  - trade-offs, 383
  - XP lifecycle and, 18–27
- Design Patterns (Gamma et al.), 314
- design spikes, 332
- designers, 27, 28, 33
  - database, 33
  - graphic, 28, 32
  - interaction, 28, 31–32, 38, 46, 47, 213
  - software, 27, 33, 312, 387
    - (see also architects)
- developers (see programmers)
- developing, 271–349
- DHH (David Heinemeier Hansson), 44
- diff debugging, 171
- distributed teams, 45, 87, 119, 258, 362
  - (see also sit together)
- divergent change, 303
- documentation, 21, 25, 195–198
  - (see also communication)
  - agile principles and, 354, 361, 370
  - code (self-documenting), 317
  - coding standards, 56
  - design, 328, 385
  - “done done” and, 156
  - phase-based organizations and, 60–61
  - product manager and, 30
  - programmers and, 33
  - reporting and, 144–151
  - requirements, 28, 60, 277, 362
    - incremental, 273
  - sitting together (communication), effects on, 110
  - spike solutions as, 331
  - stories, 256
  - tests as, 192, 285, 312
  - version control and, 171
  - vision, 30, 202
  - work-in-progress, 195
  - XP lifecycle and, 21
- domain experts (subject matter experts), 28, 31, 124
  - communicating with, 124–128, 278–284
  - customer tests and, 278–285
  - importance of, 46
  - involving, 121
  - on-site customers, 46
  - stand-up meetings and, 130
  - ubiquitous language and, 124–128
- domain knowledge/rules/logic, 31, 278
  - communicating, 28, 124–128, 278–285

- domain models, 126
- domain-driven design, 49, 126
- “done done”, 25, 156–160, 174
  - agile principles and, 379
  - bugs and, 147, 162, 164, 256, 346
  - checklist, 156
  - estimating and, 264
  - exploratory testing and, 165, 342
  - features, 214
  - iteration demo and, 143
  - mistakes and, 241
  - release planning and, 214
  - requirements and, 274
  - risk management and, 225, 229
  - slack and, 251
  - stories, 156
  - trust and, 108
  - velocity and, 234, 240, 260
- Don't Repeat Yourself (DRY), 316, 385
- drivers (pair programming), 71, 72
- DRY (Don't Repeat Yourself), 316, 385

**E**

- elaborating on a theme, 280
- emails (reporting), 145
- end-to-end tests, 297, 298
  - disadvantages of, 299
  - legacy code and, 300, 312
  - performance optimization and, 336
  - slow builds and, 180
- energized work, 20, 69, 79, 105
  - coaches and, 34
  - “done done” and, 159, 261
  - informative workspace and, 83
  - iteration planning and, 243–245, 261
  - no bugs, 162
  - overtime and, 249
  - pair programming and, 72, 77, 78
  - pressure and, 159, 244
  - slack and, 249
  - startups and, 81
  - supporting, 79
  - taking breaks, 80
  - trust and, 102, 104
  - velocity and, 261, 267
  - XP lifecycle and, 20
- engineering tasks, 235, 238–239, 243, 244
  - estimating, 263, 264
- equipment for workspaces, 52
- estimates, 33, 199, 260–270
  - consistent, 262
  - defending, 266
  - difficulty making, 265
  - doubling, 225, 231
  - explaining, 222, 266
  - improving, 267–268
  - incorrect, 142, 269
  - index cards, writing stories and, 254
  - initial, 55
  - iteration planning and, 234–237
  - learning to make, 244
  - padding, 251
  - performance optimization, 337, 338
  - planning game and, 219
  - programmers and, 33
  - release planning and, 213, 214, 219, 225, 231
  - requirements and, 274, 276
  - risk management and, 224, 231
  - scheduling, 257
  - spike solutions and, 332, 333
  - story, 234, 243, 254, 263, 274
    - size and, 255
    - success and, 4
    - tasks, 236, 244
    - technical debt and, 249
    - technical infrastructure and, 255, 257
    - Theory of Constraints and, 42, 235
    - timeboxed, 256
    - trust and, 102, 221
    - velocity and, 260, 266
    - XP lifecycle and, 19
- études, xv, 70, 100, 154, 200, 272
- evolutionary design, 321
  - (see also incremental design/architecture)
- executive sponsor, 36, 103, 204
  - involving, 52, 54, 138, 227
- existing code (see preexisting code)
- existing projects, 56–60, 227
  - (see also legacy code)
- expert customers, 273
- experts (mentors), 12
- exploratory testing, 272, 282, 341–349
  - acceptance tests and, 24
  - bugs and, 164, 167, 346
  - customer reviews and, 275
  - customer tests and, 282
  - “done done”, 157
  - end-to-end, 300

- example, 345–346
- heuristics, 343–345
- legacy code and, 59, 348
- pair programming and, 76
- producing high-quality code, 20
- test-driven development (TDD) and, 300
- testers and, 34
- XP lifecycle and, 20

exposure (risk), 227

Extreme Programming (see XP)

## F

- Facilities department, 36
- fail fast, 318, 369–370, 386
- firefighting, 92, 242
- fist, 133
- Fit (Framework for Integrated Test), 282
- FitNesse, 282
- fixtures (Fit), executing tests, 282
- flow, 71, 187
- focused integration tests, 298
- formatting (style), 133
- Fowler, Martin, xviii, 43, 316
- fractional assignment, 39
- full-time team members, 39
- function points, 146, 148
  - bugs and, 161

## G

- games (economics), 219
  - (see also planning game)
- gaming, 85
- Gamma et al., 314
- going dark (paired programming), 72
- Goldilocks heuristic, 344
- good design (software), 381–389
- graphic designers, 28, 32
- green bar (TDD), 287
- greenfield projects, 54

## H

- half-baked objects, 304
- hand-drawn charts, 84
- handoff documentation, 196
- Hansson, David Heinemeier (DHH), 44
- head (version control), 170
- heuristics, 343
- hiring (see staffing guidelines)

- Hoare, Tony, 387
- horizontal stripes, creating stories, 211
- horizontal-market software, 122
- HR (human resources) department, 36, 150
- hustle, 104

## I

- ideal engineering days (story points), 260
- impaired projects, 4
- in-house custom development, 120
- incremental change, 367
- incremental design/architecture, 321–331
  - architecture and, 324–326
  - classes and, 323–324
  - coaches and, 34–35
  - continuous design and, 322
  - cost of, 328
  - designers/architects and, 33
  - documentation and, 328
  - “done done” and, 159
  - estimating and, 269
  - iterations and, 243
  - methods and, 323
  - modeling and, 327
  - phase-based organizations and, 61
  - planning game and, 223
  - predictive and, 327
  - programmers and, 32–33
  - reflective design and, 327
  - release planning and, 218
  - simple design and, 319
  - stories and, 255, 257, 258, 269
  - XP lifecycle and, 20
- incremental requirements, 19, 32, 271–278
  - on-site customers and, 28
  - stories, 253
- index cards, 83, 92
  - archiving, 166, 239, 257
  - backups, 257
  - bugs on, 163, 166, 256
  - charters (exploratory testing) on, 342
  - contraindications, 258
  - CRC (Class, Responsibility, Collaborator), 327
  - estimates on, 262
  - incremental design with, 323–324
  - informative workspace and, 83
  - pair programming with, 73, 290–296
  - planning with, 83, 151, 213, 227, 239
  - purchasing, 53

- refactoring/to-do, 324
- retrospectives and, 226
- risk management with, 225, 227
- stakeholders and, 141, 258
- stand-up meetings, preparing with, 130
- stories on, 41, 61, 141, 163, 220, 253–257
- tasks on, 235–236, 239
- test-driven development (TDD) with, 290–296, 323
- timeboxing and, 100
- tracking with, 147, 227, 239
- trust, building with, 107
- value of, 151, 220, 253–254
- whiteboards, sticking to, 52

Industrial XP (IXP), xvii

informative workspace, 52, 69, 83–88

- charts, 84–86
- coaches and, 35
- communicating with, 131, 144, 150
- designing, 115
- energized work and, 80
- hustling, 105
- index cards, writing stories and, 254
- iteration plan, tracking in, 239
- release plan, tracking in, 213, 228
- retrospectives and, 95
- stand-up meetings, 129
- story cards and, 254
- vision and, 203
- XP lifecycle and, 19

integration

- tests, 298
- tokens, 184

integration machine (build machine), 52, 157

interaction designers, 28, 31

internal publication (interfaces), 318

inventory, 371

ISO 9001, 35, 371

iteration demos, 20, 25, 138–144

- customers and, 20
- executive sponsor and, 36, 138
- hustling, 105
- in-house custom development and, 121
- no bugs, writing, 162
- on-site customers and, 28
- product manager and, 31, 138
- stakeholders, 36
- team and, 28
- trust and, 105, 107–109

- XP lifecycle and, 20

iteration planning, whiteboard, 52, 239–240

iterations, 19, 41, 55, 199, 233–246

- adopting, 54–56, 61
- commitment, 237
- commitments, 248
- daily, 242
- demos (see iteration demos)
- “done done” and, 159
- emergencies and, 240–242
- first, 54–56
- legacy code and, 57
- length, 244–245
- lost, 240
- mistakes and, 105, 240, 248
- phase-based organizations and, 60–61
- planning, 233–245
- projects, 213
- retrospective, 91
- slack and, 246–252
- tasks, estimating, 264
- tracking, 239
- velocity and, 261
- XP lifecycle and, 18–27

## J

Jeffries, Ron, xvii, 188, 327, 360

JUnit, 296

## K

Kerth, Norm, 88, 92

keybindings when pair programming, 75

knowledge silos, 191

## L

label (version control), 170

last responsible moment, 41

- release planning and, 214, 217
- requirements and, 274

legacy code/projects, 56–60

- adopting XP and, 56
- bugs and, 167
- coding standards and, 137
- continuous integration and, 186
- exploratory testing and, 348
- iteration demos and, 141
- refactoring and, 312
- ten-minute build and, 179

- test-driven development and, 300
- Theory of Constraints and, 42

lifecycles, 18–27

local builds, 178–182

locking model (version control), 170

lost iterations, 240

## M

maintainability of software projects, 4

management

- bugs and, 166
- change and, 54, 95
- coaches and, 34
- commitments to, 54, 228–230
- pledge to, 54
- pressure from, 229–230
- reporting to, 144–148
- retrospectives and, 94
- success and, 5, 7
- support, importance of, 44
- trust and, 104–109

manifesto for Agile software development, 9

mastery, pursuing, xiii, 11, 353, 388

matrix-managed organizations, 39

mentors, 12

- coaches, contrasted with, 12
- colocated teams and, 45

merciless refactoring, 328

(see also refactoring)

merging (version control), 170

methods (process), 6, 9

- customizing (refining), 11, 353, 357
- incrementally designing, 323

mindfulness, 42

- études and, xv, 70
- mastering agility and, xiii, 353
- practicing, 70
- XP, required for, 69

mini-études, xv, 154

minimum marketable feature (MMF), 209

miracle of collaboration, 221

mitigation (risk management), 226

MMF (minimum marketable feature), 209

mock objects, 298

mock-ups, 28, 32, 274

modeling (domain), 126

Montagna, Frank, 92

multipliers (risk), 224, 230

multistage integration builds, 187

mute mapping, 93

## N

n-tier architecture, 324

navigating, 71, 75

- bugs and, 163
- first iteration, 56
- incremental design and, 323, 328
- root-cause analysis and, 89
- slack and, 246

network architect, 33

no bugs, 25, 160–169

- adopting XP and, 167
- agile principles and, 362
- communication and, 124, 163, 362
- customer reviews and, 277
- customer tests and, 162, 278
- “done done”, 158
- expectations, 166
- exploratory testing and, 24, 164, 167, 341–342, 346, 347, 348
- legacy code and, 59, 167
- novices, 166
- novices and, 161
- pair programming and, 162
- planning and, 242, 256
- programmers and, 33, 161–163
- reproducing, 171
- risk and, 225
- security and, 166
- stories, 256
- technical debt and, 40
- ten-minute build and, 182
- test-driven development and, 62, 162, 285
- testers, 164
- testers and, 34, 162
- time dependencies and, 304
- ubiquitous language and, 125, 126, 127
- XP lifecycle and, 20

no silver bullets, 3

noncolocated teams, 45, 86, 119, 258, 362

(see also sit together)

none, some, all heuristic, 344

nonfunctional/parafunctional requirements, 20, 256, 337

nulls, coddling, 304

NUnit, 296

NUnitAsp, 305

## O

- object-oriented programming/design (OOP), 126
  - static classes, 304
  - strong design skills and, 49
- OD (organizational development), 95, 109
- on-site customers, 19, 28–32, 46, 120
- once and only once (OAOO), 315, 316
- opportunity costs, 267, 371
- optimization (see performance optimization)
- oral tradition, 195
- organizational
  - adaptive planning and, 216
  - antibodies, 104
  - culture, 103, 216, 231, 363
  - development (OD), 95
  - successes, 5–7
- osmotic communication, 113
- outsourced custom development, 121
- overtime, 106, 249, 267
  - iteration length and, 244
  - productivity reduction from, 81

## P

- pair coaching, 51
- pair programming, 33, 47, 69–79, 71–79
  - challenges, 74
  - coding standards and, 135
  - collaboration, 100
  - contraindications, 77
  - energized work and, 80, 82
  - first iteration and, 56
  - how to, 72
  - management support and, 44
  - no bugs, writing, 162
  - planning sessions and, 239
  - self-documenting code, 317
  - TDD and, 287
  - team size and, 47
  - velocity and, 268
  - waste, eliminating, 368
  - XP lifecycle and, 20
- pairing stations, 52, 73, 116
- parafunctional (nonfunctional) requirements, 20, 256, 337
- Parkinson's Law, 251
- pay (compensation), 36, 44
- performance, 157
- performance optimization, 20, 25, 272, 335–340

- continuous integration and, 188
- “done done”, 158
- stories and, 254
- personal development, 120
- personal rewards of writing software, 4
- personal success, 7
- phase-based organizations, 60–61, 216
- philosophy, agile development, 9
- ping-pong pairing, 74
- planning, 199–270
  - (see also release planning, iteration planning, planning game)
- planning game, 19, 199, 219–224
  - energized work and, 80
  - iterations and, 61
  - on-site customers, 28
  - programmers and, 33
- planning horizons, 214, 217
- planning phase, 60
- position heuristic, 344
- PowerPoint presentations, 83
- practices (methods), 9, 25, 48, 354
- pragmatic idealism, 359
- predictive release planning, 218
- predictive design, 327
- preexisting code, 49
  - analyzing, 305
  - (see also legacy code)
- Pretty Adventuresome Programming, 196
- Prime Directive (Kerth), 92, 95
- primitive obsession, 303
- principles, agile, 353
  - practice, 387
- private branches (version control), 174
- process improvement charts, 84
- processes (see methods)
- product documentation, 195
- product facilitators, 202
- product managers, 27, 28, 30–31, 202
  - bugs and, 163
  - involving, 46
  - on-site, 46
  - stand-up meetings and, 130
- product vision, 201
  - documenting, 202
  - promoting, 30
- production-ready software, 156
- productivity
  - agile development, 160

- legacy code and, 56
- team size and, 39
- programmer-coaches, 35
  - (see also coaches)
- programmer-tester empathy, 103
- programmers, 27, 32–34
  - stand-up meetings, 130
  - velocity and, 267
- progress reports, 144, 145
- project community, 36
- project managers, 28
  - coaches and, 35
- project retrospectives, 91
- project-specific risks, 225
- projects, understanding, 357
- proxy customers, 46, 119
- published interfaces, 318

## Q

- quality, 161–168
  - measuring, 149
  - refactoring and, 40
  - technical debt, 267, 360
  - testers and, 34, 346
- quality assurance (QA), 20, 341, 346
  - (see also testing, testers)
- QWAN (Quality Without a Name), 381

## R

- reading groups for research, 252
- real customers, 120–124
  - feedback from, 20, 32, 203, 379
  - incremental requirements and, 273
  - iteration demos and, 138
  - on-site customers, contrasted with, 19
- “red, green, refactor” cycle, 286
- refactoring, 33, 40, 272, 303–314
  - bugs, writing, 162
  - incremental design/architecture, 321
  - pair programming and, 76
  - slack and, 247
  - TDD and, 288
  - technical debt, paying down, 248
- reflective design, 303–304
- regression tests, 20, 34, 59, 281, 347
- relationships, 99
  - (see also collaboration)
  - building effective, 102–112, 361–362
- release planning, 31, 56, 199, 206–219

- brand-new project, applying, 55
- commitments, making, 228
  - “done done”, 159
- early and often, 206
- index cards, writing stories and, 254
- iteration planning, 233
- on-site customers and, 28
- outsourced custom development and, 121
- phase-based organization and, 60
- product managers and, 31
- production-ready software, 156
- stories, 253
  - vision statements, creating, 202
- XP lifecycle and, 19
- release retrospectives, 91
- releasing software, 153–198
- reporting, 25, 144–151
  - coaching teams and, 35
  - energized work and, 80
  - hustling, 105
  - informative workspaces and, 86
  - stand-up meetings and, 131
- repositories (version control), 169, 185
- requirements, 18
- research time, 247
- retrospectives, 25, 62, 69, 91–97
  - coding standards and, 134
  - informative workspaces and, 86
  - iteration schedule and, 233
  - objective, 94
  - process improvement charts, 84
  - product managers and, 31
  - root causes and, 89
  - trust, building, 103
  - XP lifecycle and, 20
- revert (version control), 170
- Rip, Peter, 210
- risk census, 225
- risk exposure, 227
- risk management, 19, 24, 199, 224–233
  - coaches and, 35
  - estimates, making, 262
  - incremental design/architecture and, 325
  - on-site customers and, 28
  - planning projects and, 213
- risk-driven architecture, 325
- roadmaps, 145
- roles, filling, 38
- rolling back (version control), 170

- root-cause analysis, 20, 69, 88–90, 165
  - risk management and, 226
- round-trip engineering, 305
- Ruby on Rails, 44
- rules (domain) (see domain knowledge)

## S

- sandbox (version control), 169
- Satir Change Model, 53
- schedules, 91
  - deadlines and, 209, 269
  - overtime and, 81
- scopeboxed plans, 212, 231
- screechingly obvious code, 386
- Scrum, 9
  - Daily, 129
- seams (refactoring), 300
- second adopter syndrome, 54
- secondary builds, 187
- self-documenting code, 317
- self-organization, 28, 34
  - teams, 50
- shotgun surgery, 303
- silver bullets, 3
- silver stories, 252
- simple design, 61, 272, 314–321
  - bugs and, 162
  - incremental design/architecture and, 321, 325
  - pair programming and, 75
- simultaneous phases, 18
  - phase-based organizations, 60–61
- single codebase, 173
- sit together, 19, 28, 112–120
  - (see also distributed teams)
  - coaches and, 35
  - documentation and, 195
  - “done done”, 159
  - incremental requirements and, 273
  - index cards, using, 258
  - no bugs, writing, 162
  - planning sessions and, 239
  - trust, building, 103
  - velocity and, 267, 268
  - version control and, 172
  - workspaces, designing, 115
- slack, 20, 95, 199, 246–252
  - bugs, writing, 162
  - coaches and, 35
  - “done done”, 159
  - incremental requirements and, 275
  - problems, managing, 106
  - technical debt, 267
  - ten-minute build and, 181
  - velocity and, 261
- SLOC (Source Lines Of Code), 148
- slow builds, 180, 186
- smells (code), 303–304
- software, 381
- software, types of, 120–122
- source code, 384
  - (see also design)
- Source Lines Of Code (SLOC), 148
- spike solutions, 25, 172, 272, 331–335
  - driving and navigating, 73
  - estimating, 265
  - pair programming and, 76
  - performing, 331
  - slack and, 248
  - stories, 256
- stakeholders, 36, 231
- stand-up meetings, 19, 25, 61, 129–133
  - informative workspaces and, 86, 129
- static classes, 304
- status emails, 145
- stories, 41, 149, 219, 253–260
  - cards, 253–254
  - documentation and, 195
  - “done done” and, 159
  - estimating, 263–264
  - horizontal/vertical stripes, 212
  - iteration planning and, 233
  - on-site customers and, 28
  - planning, 199
  - XP lifecycle and, 19
- story points (ideal engineering days), 260
- stovepipe systems, 48
- stretch goals, 229
- strong code ownership, 194
- student syndrome, 251
- subject matter experts (domain experts), 31
- success of projects, 4
  - management support and, 44
- SUnit, 296
- supplies for workspaces, 53
- surprise retrospectives, 91
- sustaining documentation, 195
- SWAT team, 372
- synchronous integration, 186

asynchronous integration, contrasted with, 186, 188

## T

tagging (version control), 170  
taking breaks, 80  
task-switching, cost of, 39, 206  
TDD (test-driven development), 20  
teams, 27–39

- agreements, 45
- colocated, 45
- continuity, 103
- promoting, 107
- size, 38, 47–48
- trust amongst, 35

technical debt, 40

- bugs, writing, 162
- eliminating, 387
- legacy projects and, 56, 58
- slack and, 246
- velocity and, 267

technical excellence, 4, 381–389  
technical infrastructure, establishing, 55, 183, 186, 257, 321–331  
technical specialists, 33  
technical success, 7  
ten-minute build, 20, 33, 62, 141, 177–183

- automating, 177
- continuous integration and, 186
- customer tests and, 278
- “done done”, 157
- frequent releases of software and, 208
- iteration schedule, 233
- planning sessions and, 238
- technical debt, paying down, 58
- version control and, 175

test-driven development (TDD), 20, 33, 62, 272, 285–303

- coaches and, 35
- continuous integration and, 183
- documentation and, 195
- “done done”, 157, 159
- incremental design/architecture, 321
- no bugs, writing, 162
- pair programming and, 72, 76
- phase-based organization, 61
- refactoring and, 306
- slack and, 250
- technical, paying down, 58

ten-minute build and, 177  
waste, eliminating, 368  
testers, 28, 34, 59

- Theory of Constraints and, 42, 238

testing (see no bugs)

- practices in XP lifestyles, 20

testing phase (phase-based organization), 61  
testing tools (TDD), 296  
tests, customer (see customer tests)  
Theory of Constraints, 42, 372  
third-party components, 317  
throughput, 146, 371  
thumb vote, 135  
time dependencies, 304  
time usage report, 147  
timeboxed plans, 212  
timeboxing, 40, 100, 233  
tip (version control), 170  
tools

- TDD, testing, 296
- toolsets when programming in pairs, 75

transition indicators (risk management), 226  
trust, 25, 102–112

- estimating, 265
- frequent releases of software and, 208
- risk management and, 230

teams, 35

## U

ubiquitous language, 19, 25, 124, 125

- customer tests and, 283
- documentation and, 195
- programmers and, 33
- XP lifecycle and, 19

UI (user interface), 31  
UML sequence diagram, 305  
unfamiliar code, 191  
unit tests, 297  
unreleased software, 371  
updating (version control), 169, 185  
user interface (UI), 31

## V

values, 9, 353–355

- planning games, 219

velocity, 149

- estimating, 260–261
- iterations and, 61
- mapping estimates for programming, 42, 57

- slack and, 246
- version control, 20, 25, 169–177
  - continuous integration and, 186
  - incremental requirements and, 273
  - programmers and, 33
  - ten-minute build and, 177
- vertical stripes (stories), 212
- vertical-market software, 121
- visible charts, 84
- vision, 199–205, 202
  - (see also product vision)
  - documentation and, 195
  - energized work, 80
  - incremental requirements and, 274
  - statement, 145
- visionaries, 201
- VSS (Visual SourceSafe), 174

## **W**

- wannabee static class (smell), 304
- waste, eliminating, 367–373
- weak code ownership, 194
- weekly demos (see iteration demos)
- whiteboards, 83
- whole team, 114
  - (see also teams)
  - “done done”, 157
  - exploratory testing, 341
- work-in-progress documentation, 195
- working agreements, 54, 133–144
- working copies (version control), 169
- workspaces, 51
  - adopting an open, 117
  - designing, 115
  - informative, 19, 52
  - sample, 116
  - shared, using coaches, 35

## **X**

- XML, 126
- XP (Extreme Programming), 7
  - adopting, 43–63
  - lifecycles, 18–27
  - understanding, 15–42
- xUnit tools, 296

## **Y**

- YAGNI (You Aren’t Gonna Need It), 315

