



Index

A

acceptance testing, 34, 77, 145, 146
agile Manifesto, 115
Agile Software Development Ecosystems, 170
agile value stream maps, 11–13
amplifying learning principle
 feedback
 basics, 22–24
 feedback loops, 24–27
 iterations
 basics, 27–28
 convergence of development, 31
 negotiable scope of projects, 32–34
 planning, 29
 team commitment, 30
quality, 16, 17
 design cycles, 18–19
 learning cycles, 20
 right the first time approach, 19–20
 service industry, 16
 try-it, test-it, fix-it approach, 19–20
 variability of expectations, 17
set-based software development, 42–44
 versus point-based, 38–42
synchronization
 basics, 34–35

 daily build and smoke testing, 35–36
 matrix approach, 36–38
 spanning applications, 35–36

Apollo, 55
Apple, 55
applied ratio, 81
Armstrong, Lance, 155, 157
Art of the Possible, 180
Austin, Rob, 159
authorization systems, waste, 8

B

Beck, Kent, 12
Beinhocker, Eric, 55
belonging, building block of motivation, 108
Boehm, Barry
 concurrent software development, 50
 contracts, 177
Bonabeau, Eric, 64
Brooks, Fred
 master developers, 114
 team empowerment, 99
 waterfall software design process, 25
building integrity principle
 basics, 125
 conceptual integrity, 127–128, 135–137





- definition, 125
 - maintaining, 142–143
 - key factors, 128–129
 - matrix model, 132
 - model-driven design, 131–134
 - perceived integrity, 126, 129–131
 - definition, 125
 - maintaining, 134–135, 139–140
 - refactoring, 140–141
 - conceptual integrity maintenance, 142–143
 - versus* rework, 144–145
 - testing software
 - as-built systems, 148–149
 - communication, 146
 - customer testing, 145–149
 - developer testing, 145–149
 - feedback, 146–147
 - maintenance, 149
 - scaffolding, 147–148
 - burn-down charts, 33
 - business logic, 138
- C**
- Capability Maturity Model (CMM), 97, 182
 - Capability Maturity Model Integration (CMMI), 98–99, 182–183
 - centers of excellence, 122
 - Chrysler Corporation, 39–41
 - Cisco Systems, 55
 - Clark, Kim
 - conceptual integrity, 135–137
 - product integrity, 125
 - through information flow, 128
 - CMM (Capability Maturity Model), 97, 182
 - CMMI (Capability Maturity Model Integration), 98–99, 182–183
 - co-source contracts, 175
 - Cockburn, Alistair, 76
 - Collaborative Advantage*, 161
 - collective code ownership, 35–36
 - daily builds and smoke tests, 35–36
 - synchronization, 34–35
 - communities of expertise, 119–121
 - communities of scientists, 119
 - competence, building block of motivation, 109
 - conceptual domain models, 132
 - conceptual integrity, 127–128, 135–137
 - definition, 17, 125
 - maintaining, 142–143
 - concurrent development
 - product development, 47–48
 - cost escalation, 50, 52
 - software development, 48–49
 - cost escalation, 49–52
 - last responsible moment, 57–60
 - Constantine, Larry, 138
 - constraints, 82
 - contracts
 - fixed-price, 165–167
 - multistage, 169–171, 176
 - optional scope, 176–177
 - purpose of, 164–165
 - shared-benefit, 175, 177
 - target-cost, 163, 171–172, 177
 - example, 173–174
 - target-schedule, 174–175
 - time-and-materials, 167–168, 176
 - viability of trust, 161–162
 - manufacturing, 161–162
 - software, 162–164
 - Curtis, Bill, 129
 - customer testing. *See* acceptance testing
 - Customers First Program, 96
- D**
- daily build and smoke testing, 35–36
 - Death March project
 - Solution Emerges, 44–45
 - Amplifying Feedback, 27





- Eliminating Waste, 2–3
 - Weekly Iterations, 21
 - DEC, 55
 - decision making (decide as late as possible principle)
 - delaying decisions, 53–54
 - depth-first *versus* breadth-first problem solving, 60–61
 - intuitive decision making, 61–62
 - last responsible moment principle, 57–60
 - simple rules, 64–66
 - U.S. Marine Corps example, 62–64
 - delay costs
 - application models, 88–91
 - basics, 83–85
 - product models, 85–88
 - tradeoff decisions, 88–91
 - delivery (deliver as fast as possible principle)
 - basics, 69–70
 - delay costs
 - application models, 88–91
 - basics, 83–85
 - product models, 85–88
 - tradeoff decisions, 88–91
 - information radiators, 76
 - queuing theory
 - cycle time reduction, 77–81
 - queuing process, 82–83
 - reason for fast delivery, 70–71
 - scheduling for manufacturing
 - MRP (material requirements planning), 71–72
 - pull systems, 72–73
 - push systems, 71
 - scheduling for software development
 - pull systems, 74–76
 - Dell, Michael
 - contracts, 162, 164
 - inventory of computers, 9
 - Dell Computer Corporation
 - fast delivery, 70
 - last responsible moment principle, 60
 - DeMarco, Tom
 - performance measurement, 159
 - slack in organizations, 81
 - Department of Defense (DoD), 167
 - depth-first *versus* breadth-first decision making, 60–61
 - design cycles of software development, 18–19
 - deterministic controls, 25–26
 - developer testing. *See* unit testing
 - Developing Products in Half the Time*, 83
 - Domain Driven Design*, 131
 - dual ladders, leadership apprenticeship programs, 115
 - Dyer, Jeffrey, 161, 164, 171
- E**
- Eisenhardt, Kathleen, 65
 - eliminating waste principle, 1–2
 - value stream maps, 9–13
 - waste types
 - defects, 8
 - extra processes, 5–6
 - including unrequested features, 6
 - management activities, 8
 - manufacturing waste types, 4
 - motion, 7–8
 - paperwork, 5–6
 - partially done work, 5
 - task switching, 6
 - waiting, 7
 - embedded software development, 42, 185
 - empowering the team principle
 - CMM, 97
 - CMMI, 98–99
 - expertise
 - communities of expertise, 119–121
 - Nucor example, 117–118
 - standards, 121–122





Xerox example, 118–119

leadership

- fuzzy “uncertain” front end, 114
- master developers, 112–113, 115
- project management, 115–116
- respected leaders, 111–112

motivation

- building blocks, 108–109
- 3M example, 103–105
- sense of purpose, 105–107
- time expended by team members, 110–111

scientific management, 95–96

self-determination

- management improvement project, 102–103
- NUMMI project, 99–101

team polygon, 107

Evans, Eric, software architecture, 139

- model-driven, 131

expertise

- communities of expertise, 119–121
- Nucor example, 117–118
- standards, 121–122
- Xerox example, 118–119

F

FDD (Feature-Driven Development), ownership of code, 34

Federal Express, overnight delivery, 69

feedback

- basics, 22–24
- feedback loops, 24–27

Ferguson, Bruce, 175–176

“A Field Study of the Software Design Process for Large Systems,” 129

fixed-price contracts, 165–167

Ford, Henry, 95

Ford Motor Company, 95–96

Forrester, Jay, 153

Fowler, Martin, 139

Freedman, David, 62

Fry, Art, 112

Fujimoto, Takahiro

- conceptual integrity, 135–137
- product integrity, 128

G

General Electric Work-Our program, 102, 182, 184

General Motors

- contracts, 161–162
- NUMMI project, 99–101

glossaries, 132–133

Google, 126

Guindon, Raymonde, 18, 19

H

Harvard Business School, 125

Hewlett-Packard

- delaying decisions, 53–54
- options thinking, 55

Highsmith, Jim, 170

Hock, Dee, 110

Hresko, Jamie, 100

Humphrey, Watts, 97

I

IBM, 55

index cards

- versus* kanban cards, 75
- scheduling tasks, 74, 75

information radiators, 76

instructions

- large companies, 182–183
- small companies, 183–184
- special work environments, 184–185
- sphere of influence, 180–182
- troubleshooting guide, 185–186
- warranty, 186





Integra, 136
integrated problem solving, 135–137
integration testing, 145
integrity building
 basics, 125
 conceptual integrity, 127–128, 135–137
 definition, 125
 maintaining, 142–143
 key factors, 128–129
 matrix model, 132
 model-driven design, 131–134
 perceived integrity, 126, 129–131
 definition, 125
 maintaining, 134–135, 139–140
 refactoring, 140–141
 conceptual integrity maintenance,
 142–143
 versus rework, 144–145
 testing software
 as-built systems, 148–149
 communication, 146
 customer testing, 145–149
 developer testing, 145–149
 feedback, 146–147
 maintenance, 149
 scaffolding, 147–148
Intrinsic Motivation at Work, 105
intuitive decision making, 61–62
ISO9000 program, 96
 Capability Maturity Model (CMM), 97
iterations
 basics, 27–28
 convergence of development, 31
 negotiable scope of projects, 32–34
 planning, 29
 team commitment, 30
iterative software development. *See* iterations

J – K

Johnson, Jim, 177

Jones, Daniel, 9
kanban cards, 72
 versus index cards, 75
 information radiators, 76
key process areas (KPA), CMM, 98, 182
Klein, Gary, 61, 62
Kotter, John, 111
KPAs (key process areas), CMM, 98, 182

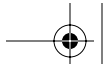
L

Larman, Craig, 139
last responsible moment in development,
 57–60
leadership
 fuzzy “uncertain” front end, 114
 master developers, 112–113, 114, 115
 project management, 115–116
 respected leaders, 111–112
Lean Construction Institute, 73
Lean Thinking, 9
lean thinking, origin, 1–2
learning cycles of software development, 20
LensCrafters, 69
Lister, Timothy, 159
L.L. Bean, 69
Lockwood, Lucy, 138

M

3M, 158–159
mapping, 138
Martin, Robert C., 139
master developers, 112–113, 114, 115
material requirements planning (MRP), 71–72
matrix approach
 matrix model, 132
 synchronization, 36–38
MBO program, 96
McBreen, Pete, 115
McCarthy, Jim, 99
McKnight, William, team motivation, 104–105





safety as building block, 108
 measurements. *See also* testing software
 basics, 155
 information, 160–161
 optimization
 local optimization, 157
 suboptimization, 155, 157–159
 performance, 159–160
Measuring and Managing Performance in Organizations, 159
 messaging, 138
 Meyer, Christopher, 64
 Microsoft
 options thinking, 55
 software development approaches, 142
 model-driven design, 131–134
 motivation
 building blocks, 108–109
 3M example, 103–105
 sense of purpose, 105–107
 time expended by team members, 110–111
 Motorola, 37
 MRP (material requirements planning), 71–72
 multistage contracts, 169–171, 176
Mythical Man Month, 99

N

New United Motor Manufacturing, Inc.
 (NUMMI) example, 99–101
 Norman, Donald, 140
 Nucor, example of expertise, 117–118
 NUMMI (New United Motor Manufacturing, Inc.) example, 99–101

O

Ocock, Tim, 170–171, 175
 Ohno, Taiichi, 1
 “one-minute scope control rule,” 3
 optimization
 local optimization, 157

suboptimization, 155, 157–159
 Optimized Operations program, 96
 optional scope contracts, 176–177
 options
 definition, 54–55
 delaying decisions, 53–54
 Microsoft example, 55
 options thinking, 52–53
 software development, 56–57
 ownership of code, synchronization, 34–35
 daily builds and smoke tests, 35–36

P

Papaccio, Philip, 177
 perceived integrity, 126, 129–131
 definition, 17, 125
 maintaining, 134–135
 guidelines, 139–140
 persistence, 138
 Petroski, Henry, 140
 P&Ls (profit and loss statements), 83, 85–87
 PMI (Project Management Institute), 183
 problem solving, depth-first *versus*
 breadth-first, 60–61
 problem solving, integrated, 135–137
Product Development Performance, 128
 production
versus development, 15–16
 lean production practices, 15
 profit-sharing contracts, 175
 progress, building block of motivation, 109
 Project Management Institute (PMI), 183
 project tracking of waste, 8
 pull scheduling
 information radiators, 76
 scheduling for manufacturing, 72–73
 scheduling for software development, 74–76
 push scheduling, 71



**Q**

- qualifiers, 133
- quality, 16, 17
 - design cycles, 18–19
 - learning cycles, 20
 - right the first time approach, 19–20
 - service industry, 16
 - try-it, test-it, fix-it approach, 19–20
 - variability of expectations, 17
- queuing theory
 - cycle time reduction, 77–81
 - queuing process, 82–83

R

- refactoring
 - basics, 140–141
 - conceptual integrity maintenance, 142–143
 - versus* rework, 144–145
- Reinertsen, Donald, 83
- right the first time software development approach, 19–20
- Royce, Winston
 - waste recognition, 4
 - waterfall software design process, 24

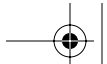
S

- safety, building block of motivation, 108
- scheduling
 - manufacturing
 - MRP (material requirements planning), 71–72
 - pull systems, 72–73
 - push systems, 71
 - software development
 - basics, 74
 - pull systems, 74–76
- scientific management, 95–96
- scope of software projects, 32–34
- Sears Catalog, 69
- seeing the whole principle

contracts

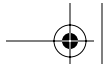
- fixed-price, 165–167
 - multistage, 169–171, 176
 - optional scope, 176–177
 - purpose of, 164–165
 - shared-benefit, 175, 177
 - target-cost, 163, 171–174, 177
 - target-schedule, 174–175
 - time-and-materials, 167–168, 176
 - viability of trust, 161–162
 - viability of trust, manufacturing, 161–162
 - viability of trust, software, 162–164
- measurements
- basics, 155
 - information, 160–161
 - optimization, 155, 157–159
 - performance, 159–160
- SEI (Software Engineering Institute), 98
- self-determination
- management improvement project, 102–103
 - NUMMI project, 99–101
- sequential software development process, 24
- Service Excellence program, 96
- set-based software development, 42–44
- versus* point-based, 38–42
- shared-benefit contracts, 175, 177
- Shingo, Shigeo, 4
- simple rules, decision making, 64–66
- Six-Sigma programs, 182
- Slack*, 81
- slack, queuing theory, 79–81
- Smith, Preston, 83
- smoke testing and daily builds, 35–36
- Sobek, Durward, 39–41
- Software Craftsmanship*, 115
- software development
- adding new features, 141–142
 - architecture basics, 137–139
 - mature, definition, 98
 - versus* production, 15–16





- simple rules, 65–66
 - Software for Use*, 138
 - Southwest Airlines, 64
 - spanning applications, 35–36
 - story cards, scheduling, 74, 75
 - “Strategy as Simple Rules,” 65
 - suboptimization, 155, 157–159
 - Sull, Donald, 65
 - Sun Microsystems, 55
 - swarm intelligence”, 64
 - synchronization
 - basics, 34–35
 - daily build and smoke testing, 35–36
 - matrix approach, 36–38
 - spanning applications, 35–36
 - system testing, 145
 - systems thinking, 153–155
- T**
- target-cost contracts, 163, 171–172, 177
 - example, 173–174
 - target-schedule contracts, 174–175
 - Taylor, Frederick, 95
 - team empowerment
 - CMM, 97
 - CMMI, 98–99
 - expertise
 - communities of expertise, 119–121
 - Nucor example, 117–118
 - standards, 121–122
 - Xerox example, 118–119
 - leadership
 - fuzzy “uncertain” front end, 114
 - master developers, 112–113, 115
 - project management, 115–116
 - respected leaders, 111–112
 - motivation
 - building blocks, 108–109
 - 3M example, 103–105
 - sense of purpose, 105–107
 - time expended by team members, 110–111
 - scientific management, 95–96
 - self-determination
 - management improvement project, 102–103
 - NUMMI project, 99–101
 - team polygon, 107
 - teamwork
 - iterations, team commitment, 30
 - velocity, 32–33
 - testing software. *See also* measurements
 - acceptance testing, 34
 - as-built systems, 148–149
 - communication, 146
 - customer testing, 145–149
 - developer testing, 145–149
 - feedback, 146–147
 - maintenance, 149
 - scaffolding, 147–148
 - try-it, test-it, fix-it approach, 19–20
 - Thimbleby, Harold, 56
 - Thomas, Kenneth, 105
 - Thompson, Fred, 168
 - thrashing, 31
 - 3M
 - accountants on product teams, 84
 - communities of expertise, 120
 - leadership in product development, 111–112
 - team motivation, 103–105
 - safety as building block, 108
 - time-and-materials contracts, 167–168, 176
 - top-down approach to development, 18
 - Total Improvement Program, 96
 - Tour de France, 155, 156
 - Toyota
 - communities of expertise, 120
 - contracts, 161–163
 - lean manufacturing, 96
 - NUMMI project, 99–101





- product development
 - approaches, 39–42
 - leadership, 111–112
 - refactoring *versus* rework, 144
- Toyota Production System
 - contracts, 162
 - refactoring, 140–141
- waste
 - eliminating waste principle, 1
 - manufacturing waste types, 4
- TQM program, 96
- traditional value stream maps, 10
- transaction management, 138
- try-it, test-it, fix-it software development approach, 19–20

U

- unit testing, 77, 145, 146
- U.S. Department of Defense (DoD), 167
- U.S. Marine Corps, 62–64
- U.S. Postal Service, overnight delivery, 69
- use case models, 133
- user interface, 138

V

- value stream maps
 - agile, 11–13
 - basics, 9, 12
 - mapping processes, 9–10
 - traditional, 10
- velocity, 32–33
- Visa, team motivation, 110

W

- waste
 - eliminating
 - value stream maps, 9–13
 - waste elimination principle, 1–2
 - types
 - defects, 8
 - extra processes, 5–6
 - including unrequested features, 6
 - management activities, 8
 - manufacturing waste types, 4
 - motion, 7–8
 - paperwork, 5–6
 - partially done work, 5
 - task switching, 6
 - waiting, 7
- waterfall software development process, 24–27
- “What Leaders Really Do,” 111
- Wheeler, Earl, 99
- whys (five), 154
- Wolfowitz, Paul, 183
- Womack, James
 - fast delivery, 70
 - value stream mapping, 9
- Work-Our program, General Electric, 102, 182, 184
- wrappers, 138

X – Z

- Xerox, example of expertise, 118–119
- Zaninotto, Enrico, 53–54
- Zero Defects program, 96

