Index

3D modeling, 118
3M
  example, 213–214
  feasibility stage, 47
  focus on the job, 52
  handspreads, 47
  samples of new products, 47
7 principles. *See* seven principles.
7 wastes. *See* seven wastes.
14 points of Deming, 122–123
80/20 rule, 25–26

A

A3 reports, 157–158
acceptance tests, 150, 186. *See also* story tests.
accommodations, 233
accountability, 64–65
Aden, Jill, 195
adopting new technologies, 230–231
agile software development, tools for. *See* Rally.
Airline Information Management System, 119
airport check-in desk example, 110
airport delays, example, 136–137
Aisin fire, 208–209, 211
AJAX, 150
Alias, 55
alignment, 69
allegiance, 214–216
Allen, Charles, 234
American auto industry, 2–3
American System of Manufacture, 1
analyzing the situation, 169
andon, 139–140
annual performance rating. *See* performance evaluations.
applause, 210
Appleton, Brad, 202
approval process, 84, 103
architecture, software
  definition, 20
  divisible systems, 182
  feedback and quality, 182
assemble-to-order, 34
assembly line. *See* mass production.
assessment, 188–189
asynchronous cadence, 109
Austin, Rob, 40
auto industry. *See also* specific industries.
  America, 2–3
  Japan, 4–7
  used car sales, 41
Autodesk, 55
automating complexity, 72–73
automating routine tasks, 197–198, 227–228, 231–232
autonomation (Jidoka), 5–6
availability of processes, 98

B

BAA airport management, 217–218, 220–221
backlog items, 185–186
balanced scorecards, 144
barriers
  eliminating, 210
  interdepartmental, 122
batch and queue approach, 88
Beck, Kent, xx
Bell, Gordon, 165
bell curve, and individual performance, 130
Bell Laboratories, 121
Benneton, 67
Beyond the Goal, 230
big visible charts, 140
billing system example, 167–168
Black Belts, 229
blame, 143
Blanc, Honoré, 1–2
Blenko, Marcia, 57
BMI, 39
BMI call center, outsourcing, 215
Boehm, Barry, 33

Boeing
777 project, 117–120, 140, 230
787 Dreamliner, 210
outsourcing, 216–217
Bohnet, Ralph, 167
bonuses as incentives, 145, 146
books and publications
Beyond the Goal, 230
Cheaper by the Dozen, 37
"Collaboration Rules," 208
Conquering Complexity in Your Business, 67
"Do You Have Too Much IT?", 69
Estimating and Planning, 232–233
Fit for Developing Software, 187
Hidden Value, 146
The Instructor, the Man and the Job, 234
The Knowledge-Creating Company, 156
Lean Software Development: An Agile Toolkit, xxiii
Lean Solutions, 43
The Living Company, 141, 225
The Machine That Changed the World, 11
Managing the Unexpected, 9
Product Development for the Lean Enterprise, 15
Product Development Performance, 13, 52
"Quality With a Name," 20
Taxonomy of Problem Management Activities, 20
Toyota Production System, 5
The Toyota Way, 14
The Ultimate Question, 241
"When IT's Customers Are External," 62–63
Who has the D?, 57
Working Effectively with Legacy Code, 167
bottleneck elimination, xix
bottlenecks (Muri), xix
boundaries, organizational
cascading queues, 113–114
cost of crossing, 39–40, 243
lean supply chains, 13
relational contracts, 221
teams, 214
value streams, 84
boundaries, system, 201
Brin, Sergey, 46
building quality in, 25–29
burn-down charts, 140
business case, 240
business intent, testing, 200
business process, 17, 20, 181
business success
constraints, 153
responsibility for, 13, 16, 53
rewards for, 145

C

cadence
asynchronous, example, 109
cycle time reduction, 108–109
establishing, 108–109
Cagan, Martin, 53
Canada, 231–232
capable development process, 98
capacity, limiting work to
cycle time reduction, 110–111
teams, 134
cascading queues, 113–114
cash stage, 49
cause. *See* root causes.
champions, 52–57, 133
change
agents, 229
management, 25
scope bloat, 25
scope control, 25
tolerance, 182
waste, 25
change for the better (Kaizen) events, 173–175
change requests, 62
chartering teams, 241
charts, 140
Cheaper by the Dozen, 37
chief architect, 133
chief engineer, 53–55
Christensen, Clayton, 226
Chrysler
NS minivan, 56
QFD (quality function deployment) analysis, 56
shared leadership, 56
churn
requirements, 24, 91
test-and-fix, 24
value streams, 91
waste, 24
Clark, Kim B., 52
Clark, Mike, 197
ClearStream Consulting, 167–168
Cleland-Huang, Jane, 182
CMM, 124
coaches, 133

code
  complexity, 69
  source of waste, 74–75
  technical debt, 150
  undeployed, 75
  undocumented, 75
  unsynchronized, 74
  untested, 74
code reviews, 194–195
coffee cup simulation, 10–11
Cohn, Mike, 232
collaboration. *See* partners; teams.
"Collaboration Rules," 208
co-located teams, 211, 213
commitment. *See also* Just-in-Time commitment.
  to change, 151
  deferring, principle of, 32–33
  iterative development, 186
  planning as, 33
committers, 209–210
companies
  life expectancy, 225–227
  organizational boundaries
    cascading queues, 113–114
    cost of crossing, 39–40, 243
    lean supply chains, 13
    relational contracts, 221
    teams, 214
    value streams, 84
  purpose of, 123
  types of, 141
compensation
  alternatives to money, 145–146
  annual raises, 144
  balanced scorecards, 144
  bonuses, 145, 146
  promotion systems, 143–144
  reward basis, 144–145
  span of influence *versus* span of control, 144–145
competing on the basis of time, 34
competitive advantage
  complexity, 69
  customer satisfaction, 241
  development speed, 35
  expert workforce, 37
  feedback, 177
  lean principles, 11
  management innovation, 124
  outsourcing, 215–216
  Toyota, 224
  user interface, 189
complete teams, 57–60
complexity
  automating, 72–73
  competitive advantage, 69
  cost of, 69–70
  limiting features and functions, 70–71
  minimum useful feature sets, 71–72
  pricing structure, example, 72–73
  prioritizing features, 71–72
  root cause of waste, 67
  software code, 69
concept stage, 46
concurrent development, 182
concurrent engineering, 16
condensing knowledge, 157
configuration management, 201–202
conflict of interest, 215
conquering complexity, 5
Conquering Complexity in Your Business, 67
constraints, 230–233
continuous improvement
  cadence, 168
  complexity reduction, 166
  configuration management, 201
  Deming's 14 points, 122
  development organization objectives, 239
  at PatientKeeper, 98
  principle of, 38
  waste elimination, 166
continuous integration, 202–203
contractors, 218
contracts
  BAA airport management, 217–218, 220–221
  fixed price, 125
  Norwegian Computer Society, 218–219
  NTNU (Norwegian University of Science and Technology), 218–219
  PS 2000, 218–219
  purpose of, 217
  relational, 219–221
  T5 Agreement, 217–218
  time and materials, 218
Cook, Scott, 51, 55
costs
  competing on the basis of time, 34
  complexity, 24–25, 69–70
  crossing organizational boundaries, 39–40, 243
  economies of scale, 5
  extra features, 24–25
  joint ventures, 220
  Keiretsu advantage, 12

costs *(continued)*:
lifecycle, 20, 70–71
refactoring, 166
of software maintenance, 20–21
standards, 193
support and warranty, 164
synergistic relationships, 221
target, 180, 218–219, 221
counterintuitive concepts
continuous integration, 202
Lean, 11
new paradigms, 11
object orientation, 195
set based development, 161
seven principles, 23
Crawford-Mason, Clare, 125
create knowledge, principle of, 29–32
Critical Chain, 232–233
cross-functional teams, 56, 64, 78, 122
Cunningham, Ward, 187
custom systems. *See* software development, custom systems.
customer-focused organizations
champions, 52–57
chief engineer, 53–55
complete teams, 57–60
decision making, 57
designing for manufacturability, 58–59
designing for operations, 58–59
development goal, 55
facilitating information flow, 52–60
leadership, 52–57
leadership teams, 55
Murphy's Law, 59–60
responsibility, 56–57
shared leadership, 56
What can go wrong, will go wrong, 59–60
customers
delighting, 49–52. *See also* Google.
focus on the job, 51–52
Kano model, 49–52
needs, 43
satisfaction, 49–52
satisfaction, as competitive advantage, 241
satisfaction, measurements, 241
service, example, 111–112
understanding, 50
cycle time
measurements, 238–240
PatientKeeper, 97–98
reducing
establishing a cadence, 108–109
evening out work arrival, 103–105
limiting work to capacity, 110–111
minimizing process elements, 105–107
minimizing process size, 107–108
pull scheduling, 112–114
speed, 98–99
utilization and, 102

D

Darwin Information Typing Architecture (DITA), 131
dashboards, 136, 140–141
de Geus, Arie, 141, 225
decisions. *See also* commitment.
irreversible, 160
key, 162
making, 57
decomposition, optimizing by, 40–41
defects
discovering *versus* preventing, 27, 82. *See also* test-driven development.
inspecting for, 27, 82
as management problems, 29
queues, 25–26
rates, 27, 34, 81, 85
seven wastes, 81–82
tracking systems, 27
defer commitment, principle of, 32–33
delays
mapping in value streams, 91
seven wastes, 80–81
delighters, 65
delighting customers, 49–52
deliver fast, principle of, 34–35. *See also* speed.
Dell Computer, 11–13
Deming, W. Edwards
14 points
overview, 122–123
point 12, 210
points 6 and 7, 210
causes of problems, 121, 123–124
choosing suppliers, 122, 123
Deming Cycle, 121
dependence on inspection, 122
fear, 122
inherent system variation, 121
interdepartmental barriers, 122
introduction, 120
leadership, 122
numerical quotas, 123
PDCA (plan, do, check, act), 121, 154–155
pride of workmanship, 123

psychology, 122
purpose of a company, 123
scientific method, 121
slogans, exhortations, and targets, 123
synergy, 121
System of Profound Knowledge, 121
theory of knowledge, 121
training, 122, 123
Deming Cycle, 121
democracy principle, Google, 45
Denne, Mark, 182
dependencies, teams, 135
deployment
available to production, 87, 90
average time, 6, 86
concept-to-launch time, 99, 103
cycle time, 170, 238–239
delays, 91
incremental, 178–179
minimum useful feature sets, 71
obsolescence, 91
Polaris project, 178–179
QFD (quality function deployment) analysis, 56
undeployed code, 75
design
of code. *See* software development.
intent, testing, 200
for manufacturability, 58–59
for operations, 58–59
of products. *See* Toyota Product Development System; Toyota Production System.
Design for Six Sigma (DFSS), 229
design/build teams, 118, 123, 133
deskilling, 228
deterministic school, 21
detractor, 65, 241
Detroit, 2, 4, 117
developing software. *See* software development.
development teams
3M, 56–60
capacity, 99
champions, 132
DFSS (Design for Six Sigma), 229
error prevention, 82
expertise, 129–130, 212
goal of, 240
incentives, 123
interaction designers, 189
joined at the hip, 55
maintenance duties, 79
measurements, 237
pride in workmanship, 210
process improvement, 31
pull scheduling, 112–114
rewards, 145
set-based concurrent engineering, 16
size, and technical debt, 153
DFSS (Design for Six Sigma), 229
differentiation, 50
discipline
automating routine tasks, 197–198
code reviews, 194–195
configuration management, 201–202
continuous integration, 202–203
five S's, 190–192
merging subsystems, 203–204
mistake-proofing, 196–198
nested synchronization, 203–204
Open Source reviews, 196
organizing a workspace, 190–192
pairing, 195–196
shine (seiso), 191–192
sort (seiri), 191–192
standardize (seiketsu), 191–192
standards for software development, 193–196
sustain (shitsuke), 191–192
systematize (seiton), 191–192
test-driven development, 198–201
dispatching, 137–138
DITA (Darwin Information Typing Architecture), 131
divisible systems architecture, 182
Do It Right the First Time, 165
do it right the first time, 29
"Do You Have Too Much IT?", 69
doctor's appointments, example, 104–105
documentation, 74, 77
domain, 82, 180, 183
domain models, 185–186
Drucker, Peter, 12–13, 220–221
dual ladder, 143
dysfunctional measurements, 238

E

Easel Corporation, xvii
economic companies, 141
economies of scale, 4, 68
education. *See* training.
80/20 rule, 25–26
eliminate waste, principle of, 23–25
eliminating barriers, 210
embedded software, 20, 163
empirical school, 21

employees. *See* partners; people; teams.
engaged thinking people, 35, 37, 117, 237
enterprise software, 20
entrepreneurial leaders, 16, 37, 54
ERP (Enterprise Resource Planning), 231
estimates
  as commitments, 232
  granular level, 134
  implementation effort, 185
  stories, 183
  tasks, 97
Estimating and Planning, 232–233
Evans, Eric, 186
Evans, Phillip, 208
Excel, 36
excellence principle, Google, 45
exchanging tests, 212
exhortations, 123
exhortations as incentives, 123
expediting projects, 98
experimentation, 171–172
expert technical workforce, 37
expertise, in teams, 129–131
exploratory tests, 201
extra features, as waste, 24–25, 75

F

FAA (Federal Aviation Administration), 119
face-to-face discussion, 78
fail fast, 118–119
fast delivery. *See* deliver fast; speed.
fear as incentive, 122
feasibility stage, 46–47
Feathers, Michael, 167
features
  limiting, 70–71, 165
  minimum useful sets, 71–72
  prioritizing, 71–72
  wastes, 24–25, 75
  YAGNI (You Aren't Going to Need It), 165
FedEx, 34
feedback, and quality
  architecture, 182
  competitive advantage, 177
  iterative development, 183–190
  Polaris program, 177–182
  release planning, 179–181
financial results. *See* return on investment.
fire, Aisin plant, 208–209, 211
FIT (Framework for Integrated Tests), 75, 150, 187
Fit for Developing Software, 187
Fitnesse, 150
five S's, 190–192
fixed price contracts, 125
fixtures, 187
focus on the job, 51–52
Ford, Henry, 2–3
Ford Motor Company, 2–3
14 points of Deming, 122–123
Fowler, Martin, 167
framework for integrated tests. *See* FIT (Framework for Integrated Tests).
France, 1–2
Francis, Charles A., 3
frequent integration, 212
Fujimoto, Takahiro, 52
Fujitsu, 39
funding profiles, 61
future blindness, 226

G

games, 17, 48, 181
Gap, 68
Gates, Bill, 36
GE Workout, 173–175
genchi-genbutsu (go, see, confirm), 54
General Motors, 2–3
George, Michael, 67
Gilbreth, Frank, 37–38
Gilbreth, Lillian, 37–38
global networks, 210–214
global teams, 212
global work groups, 212
goal setting, 223
Goldratt, Eliyahu, 230, 232
Google
  corporate philosophy, 44
  customer satisfaction, 50
  democracy principle, 45
  excellence principle, 45
  history of, 43–44
  Keyhole, 45
  maps, 45
  page rank system, 48
  product development principles, 44–45
  product development timeline
    cash stage, 49
    concept stage, 46
    feasibility stage, 46–47
    pilot stage, 48
    systems design stage, 47
  queuing theory, 101–102
  speed principle, 45

startup, 46–47
value principle, 44
workforce utilization, 101–102
Google Earth, 45
Google Local, 45
Green Book, 6

H

hack-a-thon, 152
hacking *versus* speed, 35
Hamel, Gary, 117, 124–125
handoffs, 77–78
hangers, theft of, 125
hardening software, 150–151
haste makes waste, 35
Heathrow, 217
help desk, BMI, 39
help each other, 35, 127, 129, 183
Hidden Value, 146
history of lean software development
See Just-in-Time
See mass production
See Toyota Product Development System
See Toyota Production System
H&M, 67
Honda, xxiii, 55
Honeywell, 119–120
HTTPUnit, 150
hypothesis development, 171, 234–241

I

IBM AT cables, 196–198
incentives
applause, 210
blame, 143
individual performance, 142
performance evaluations, 141–143
rankings, 142–143
incremental development, dangers of, 164
incremental funding, 61
Inditex, 67, 69
individual performance as incentives, 142
industrial model, 2, 5, 11
infrastructure, outsourcing, 214–215
innovation
management, 124, 218
start of, 46
Web inspired, 233
inspections
dependence on, 122
discovering defects, 27, 82. *See also* test-driven development.
preventing defects, 27, 82. *See also* test-driven development.
purpose of, 27
types of, 27
The Instructor, the Man and the Job, 234
integration
continuous, 202–203
frequent, 212
interaction designers, 55, 130, 189
interchangeable parts, 1–2
interchangeable people, 2–3
interdepartmental barriers, 122
Internet age, and knowledge creation, 159
intrinsic rewards, 146
Intuit
complete teams, 57–58
founding of, 51
leadership teams, 55
limiting complexity, 70
QuickBooks, 70
Quicken
introduction of, 51
leadership teams, 55
Quicken Rental Property Management, 57–58
inventory. *See also* Just-in-Time.
coffee cup simulation, 10–11
pull system, 10–11
rocks-and-stream metaphor, 7–8
as waste, 24
irreversible decisions, 160
ISO 9000, 124–125
IT departments
accountability, 64–65
business collaboration, 62–65
cost, 68
external customers, 62–63
fixing, 64
guide to the use of technology, 69
versus software companies, 62–65
we-they model, 63
workload example, 103–104
iterative development
assessment, 188–189
commitment, 186
example, 184
feedback and quality, 183–190
FIT (Framework for Integrated Tests), 187
implementation, 186–188
introduction, 183–184

iterative development *(continued)*:
overview, 183
planning, 186
preparation, 185–186
stories, 183–186
story-test driven development, 186
user interface variation, 189–190

J
Japan. *See also* Toyota; Toyota Product Development System; Toyota Production System.
auto industry, 4–7
textile industry, 3–4
Java, five S's, 192
Jefferson, Thomas, 1
Jensen, Bent, 80
Jidoka (autonomation), 5–6
JIFFIE, 151
job grades, 143–144
Job Instruction (JI) module, 235–236
Job Methods (JM) module, 235–236
Job Relations (JR) module, 235–236
Johnson, Jim, 24
joined at the hip, 55
joint ventures, 220–221
Jones, Daniel, 43
journey
accommodations, 233
adopting new technologies, 230–231
automating routine tasks, 227–228, 231–232
centering on people, 227–228
corporate life expectancy, 225–227
Critical Chain, 232–233
developing a hypothesis, 234–241
ERP (Enterprise Resource Planning), 231
future blindness, 226
goal setting, 223
measurement, 237–241
push *versus* pull systems, 236–237
right to think, 237
road map, 242
schedules, 228
Six Sigma, 229–230
Theory of Constraints, 230–233
thinking, 236–237
tools *versus* results, 229–230
training, 234–236
the use of technology, 227–228
JR (Job Relations) module, 235–236
Jula, John, 54
junior people, 130–131, 144
JUnit, 150
Juran, J. M., 26
Just-in-Time. *See also* inventory.
autonomation (Jidoka), 5–6
definition, 4
Green Book, 6
Just-in-Time flow, 5
maximizing local efficiencies, 8
mistake-proof systems, 6–7
nonstock production, 6
rocks-and-stream metaphor, 7–8
stop-the-line culture, 5–6
zero inspection, 6–7
Just-in-Time commitment. *See also* commitment.
dangers of incremental development, 164
Do It Right the First Time, 165
example, 167–168
examples
medical device interface, 162
pluggable interfaces, 163
red-eye reduction, 162–163
introduction, 159–160
irreversible decisions, 160
key decisions, 162
legacy systems, 166–168
refactoring, 164–168
and scientific method, 154
set-based design, 160–164
and waste, 164
YAGNI (You Aren't Going to Need It), 165
Just-in-Time manufacturing, 4–7

K
Kaizen (change for the better) events, 173–175
Kanban, 10–11, 136, 138–139
kanban cards, 10–11
Kano, Noriaki, 49–52
Kano model, 49–52
Keiretsu, 12–13
Kennedy, Michael, 15
key decisions, 162
Keyhole, 45
knowledge
creation
A3 reports, 157–158
condensing knowledge, 157
in the Internet age, 159
keeping notebooks, 156–157
lost knowledge, 155–159
principle of, 29–32
problem definition, 152–153
at Rally Software Development, 149–152

technical debt, 150
tracking knowledge, 155–159
theory of, 121
knowledge-based engineering, 15
The Knowledge-Creating Company, 156

L

large group improvement, 173–175
large-batch software development, 71, 102
last responsible moment, 32, 161, 185
lava lamp, 140, 198
leadership
customer-focused organizations, 52–57
Deming's points, 122
entrepreneurial, 16, 37, 54
Honda, 55
Intuit, 55
Open Source, 209–210
process, 132–133
Strong Project Leader, 54
teams, 55, 132–133
technical, 132–133
traveling team leaders, 213
lean
definition, xxiii
initiatives
first step, 153
initiating. *See* journey.
reasons for failure, 153
manufacturing
versus development, 14
overview, 11–12
principles, competitive advantage, 11. *See also* seven principles.
production
See also lean, software development
See also mass production
See also Toyota Product Development System
See also Toyota Production System
Dell Computer, 11–13
flowchart, 12
Keiretsu, 12–13
knowledge-based engineering, 15
manufacturing, 11–12
manufacturing *versus* development, 14
operations, 11–12
product development, 13–15
Southwest Airlines, 11–12
supply chain, 12–13
Toyota *versus* other vehicle manufacturers, 13
software development
history of
See lean, production
See mass production
See Toyota Product Development System
See Toyota Production System
overview, 17
Lean Solutions, 43
learn-by-doing, 19
learning. *See* training.
legacy systems, 166–168
Lexus, 13
lifecycle costs, 20, 70–71
Liker, Jeffrey, 14
limiting work to capacity, 110–111, 134
Linux security breach, example, 207–208, 211
Little's Law, 100–101
The Living Company, 141, 225
L.L. Bean, 34
local efficiencies, 8
looms, automated, 3–4
lost knowledge, 155–159

M

MacCormack, Alan, 30
MacGibbon, Simon, 62
The Machine That Changed the World, 11
maintenance
cost of, 20–21
staffing for, 79–80
management
functional, 133
innovation as competitive advantage, 124
people. *See* people, managing.
project, 133. *See also* project managers.
Managing the Unexpected, 9
manufacturing. *See also* Toyota Product Development System; Toyota Production System.
versus development, 14
Just-in-Time, 4–7
lean, 14
lean production, 11–12
mass production, 12–13
video cassettes, 59
mapping value streams. *See* value streams.
maps, Google, 45
Marick, Brian, 166, 199
market research, 56, 62–63
market share, 61, 241
Martens, Ryan, 149

mass production
See also lean, production
See also Toyota Product Development System
See also Toyota Production System
American auto industry, 2–3
American System of Manufacture, 1
Ford Motor Company, 2–3
General Motors, 2–3
interchangeable parts, 1–2
interchangeable people, 2–3
Japanese auto industry, 4–7
Japanese textiles, 3–4
Just-in-Time manufacturing, 4–7
and lean manufacturing, 12–13
maximizing local efficiencies, 8
McAfee, Andrew, 69
McCabe Cyclomatic Complexity Index, 194–195
Measure UP, 40–41
measurements
customer satisfaction, 241
cycle time, 238–240
decreasing number of, 40–41
dysfunctional, 238
improving the wrong ones, 237
Measure UP, 40–41
net promoter score, 241
optimize by decomposition, 40–41
raising levels, 40–41
reducing the number of, 238
ROI (return on investment), 240–241
Sloan, Alfred P., 40–41
Sloan's metrics, 40–41
statistical process control, 120–122
medical device interface example, 162
Meszaros, Gerard, 167
MetaScrum meeting, xvii
metrics. *See* measurements.
Microsoft, respect for people, 36
Miller, Lynn, 55, 189
mindfulness, 9
mind-meld, 50
minimum useful feature sets, 71–72
Minoura, Teruyuki, 236
mistake-proofing, 6–7, 196–198
money, as incentive, 145–146
Muda (waste), xix
Mugridge, Rick, 187
Mulally, Alan, 118, 123, 140
multitasking, causing waste, 78–80
Mura (stress), xix
Muri (bottlenecks), xix
Murphy's Law, 59–60
myths
finishing the code, 79
haste makes waste, 35
one best way, 37–38
optimize by decomposition, 40–41
planning is commitment, 33
predictable outcomes, 31–32
specifications reduce waste, 24–25
testing to find defects, 28–29

N

National Center for Manufacturing Sciences (NCMS), 13
nested synchronization, 203–204
net promoter score, 241
New United Motor Manufacturing Incorporated (NUMMI), 226
newspaper, online subscription, 50
no partial credit, 188
no secrets, 118
Nonaka, Ikujiro, 156
nonfunctional requirements, testing, 201
nonstock production, 6
non-value-added waste, 23, 83
Norwegian Computer Society, 218–219
Norwegian University of Science and Technology (NTNU), 218–219
notebooks, keeping, 156–157
NS minivan, 56
numerical quotas, 123

O

Ohno, Taiichi
introduction, 5–6
planning, 33
value streams, 83
waste, 23–25, 75
on the job training, 234–236
one best way, 2, 37–38
one click build, 198
Oobeya, 213
Open Source
chief engineer approach, 54
leadership, 54
reviews, 196
software example, 209–210
Strong Project Leader, 54
operations, lean, 11–12
optimize by decomposition, 40–41
optimize the whole, principle of, 38– 41

options-based development, 135
ordinary employees, 117, 227
O'Reilly, Charles, 146
organizational boundaries. *See* boundaries, organizational.
organizing a workspace, 190–192
organizing work, 138–139
outsourcing
  basic principles, 216–217
  BMI call center, 215
  Boeing, 216–217
  competitive advantage, 215–216
  conflict of interest, 215
  development, 216–217
  infrastructure, 214–215
  introduction, 214
  Procter & Gamble, 216–217
  Toyota, 216–217
  transactions, 215
overproduction, 25, 75
overtime, 110–111

P

Page, Larry, 46
page rank system, Google, 48
pairing, 195–196
Pareto analysis, 26
partially done work, 74–75
partners. *See also* teams.
  committers, 209–210
  contracts
    BAA airport management, 217–218, 220–221
    Norwegian Computer Society, 218–219
    NTNU (Norwegian University of Science and Technology), 218–219
    PS 2000, 218–219
    purpose of, 217
    relational, 219–221
    T5 Agreement, 217–218
  Deming point 12, 210
  eliminating barriers, 210
  equality of, 213
  examples
    3M, 213–214
    Boeing 787 Dreamliner, 210
    Linux security breach, 207–208, 211
    Open Source software, 209–210
    Procter & Gamble, 210
  exchanging tests, 212
  frequent integration, 212
  global networks, 210–214
  global teams, 212
  global work groups, 212
  joint ventures, 220–221
  leaders, 209–210
  Oobeya, 213
  outsourcing
    basic principles, 216–217
    BMI call center, 215
    Boeing, 216–217
    development, 216–217
    infrastructure, 214–215
    introduction, 214
    Procter & Gamble, 216–217
    Toyota, 216–217
    transactions, 215
  proxies, 213
  rotating people, 212
  synergy, 207–217
  traveling team leaders, 213
  war room, 213
PatientKeeper
  cycle time, 97–98
  delivery speed, 95–98
  development teams, 97
  introduction of Scrum, xvii
  limiting complexity, 71
  limiting work to capacity, 134
  product managers, 97
  release schedules, 97
PBS documentary, 119
PDCA (plan, do, check, act), 121, 154–155
people, managing
  andon, 139–140
  under the bell curve, 130
  Boeing 777 project, 117–120, 140
  causes of low quality and productivity, 121
  centering on people, 227–228
  choosing suppliers, 122
  compensation
    alternatives to money, 145–146
    annual raises, 144
    balanced scorecards, 144
    bonuses, 145, 146
    promotion systems, 143–144
    reward basis, 144–145
    span of influence *versus* span of control, 144–145
  dashboards, 136, 140–141
  Deming Cycle, 121
  Deming on, 120–123
  dependence on inspection, 122
  fear, 122

people, managing *(continued)*:
incentives
individual performance, 142
performance evaluations, 141–143
rankings, 142–143
inherent system variation, 121
interdepartmental barriers, 122
job grades, 143–144
junior people, 130–131, 144
kanban, 136, 138–139
leadership, 122
numerical quotas, 123
ordinary employees, 117, 227
organizing work, 138–139
PBS documentary, 119
PDCA (plan, do, check, act), 121, 154–155
pride of workmanship, 123
projects *versus* products, 62
psychology, 122
rotating assignments, 212
scientific method, 121
self-directing work, 137–141
sharing early and often, 118
slogans, exhortations, and targets, 123
stop-the-line culture, 139–140
synergy, 121
System of Profound Knowledge, 121
testing early, failing fast, 118–119
theory of knowledge, 121
training, 122, 123, 129
trust, 125
visible signals, 139–140
visual workspace, 136–141
wall charts, 140
why programs fail, 124–125
Working Together program, 118–120
performance evaluations as incentives, 141–143
personnel. *See* partners; people; teams.
PERT (Program Evaluation and Review Technique), 179
Pfeffer, Jeffrey, 146
pilot stage, 48
P&L (profit and loss) model, 240
plan, do, check, act (PDCA), 121, 154–155
plan-driven methods, 33
planning
as commitment, 33
iterative development, 186
Taiichi Ohno on, 33
pluggable interfaces example, 163
Polaris program, 177–182
policies. *See* practices; principles.
Post-it Notes, 139
practices. *See also* principles.
definition, 19
for successful software development, 30
predictable outcomes, 31–32
Price, Jerry, 125
pricing structure, complexity example, 72–73
pride of workmanship, 123
principles. *See also* practices.
continuous improvement, 38
definition, 19
Google
democracy principle, 45
excellence principle, 45
product development principles, 44–45
speed principle, 45
value principle, 44
lean software development. *See* seven principles.
learn-by-doing, 19
of outsourcing, 216–217
software development, 20–21
understand-before-doing, 19
prioritizing features, 71–72
Prius, 21
problem solving
analyzing the situation, 169
defining the problem, 152–153, 169
disciplined approach, 169–172
experimentation, 171–172
first rule, 168
follow up, 172
hypothesis generation, 171
introduction, 168
Kaizen (change for the better) events, 173–175
large group improvement, 173–175
scientific method, 154, 169–172
standardization, 172
verifying results, 172
process cycle efficiency, 85–86, 90–92, 108
process leadership, 132–133
processes
availability, 98
average time, calculating, 100–101
capable, 98
minimizing elements, 105–107
minimizing size, 107–108
quality measurement, 99
robust, 177
too big, 107–108
too many things, 105–107

Procter & Gamble, 51, 210, 216–217
product development, lean, 13–15
Product Development for the Lean Enterprise, 15
Product Development Performance, 13, 52
product managers, 133
product owners, 133
productivity, 28
products
  concept stage, 46
  development. *See* software development; Toyota Product Development System; Toyota Production System.
  versus projects, 60–63
  specifications
    basis for acceptance tests, 150
    waste reduction, 24–25
profit, definition, 152
profit and loss (P&L) model, 240
profitability, 61, 122, 241–242
Program Evaluation and Review Technique (PERT), 179
programmer tests. *See* unit tests.
programmers. *See* partners; people; teams.
project managers, 42, 127, 133, 237. *See also* management.
projects
  average process time, calculating, 100–101
  average speed, 99–100
  cycle time, 98–99
  dividing work into stories, 99
  expediting, 98
  measuring, 99
  PatientKeeper delivery cycle, 95–98
  process availability, 98
  process capability, 98
  versus products, 60–63
  red flags, 98
  setting upper limits, 99
  setting upper size limits, 99
  time delays, 98–99
promotion systems as incentives, 143–144
property tests, 201
Proulx, Tom, 55
proxies, 213
PS 2000 contract, 218–219
psychology, 122
pull scheduling, example, 112–113
pull systems, 10–11, 236–237
push systems, 236–237

Q

QA (Quality Assurance), 89, 96
QFD (quality function deployment) analysis, 56
quality
  building in, principle of, 25–29
  change tolerance, 182
  discipline
    automating routine tasks, 197–198
    code reviews, 194–195
    configuration management, 201–202
    continuous integration, 202–203
    five S's, 190–192
    merging subsystems, 203–204
    mistake-proofing, 196–198
    nested synchronization, 203–204
    Open Source reviews, 196
    organizing a workspace, 190–192
    pairing, 195–196
    shine (seiso), 191–192
    sort (seiri), 191–192
    standardize (seiketsu), 191–192
    standards for software development, 193–196
    sustain (shitsuke), 191–192
    systematize (seiton), 191–192
    test-driven development, 198–201
  divisible systems architecture, 182
  iterative development
    assessment, 188–189
    commitment, 186
    example, 184
    FIT (Framework for Integrated Tests), 187
    implementation, 186–188
    introduction, 183–184
    overview, 183
    planning, 186
    preparation, 185–186
    stories, 183–186
    story-test driven development, 186
    user interface variation, 189–190
  robust development processes, 177
  role of feedback
    architecture, 182
    iterative development, 183–190
    Polaris program, 177–182
    release planning, 179–181
Quality Assurance (QA), 89, 96
quality function deployment (QFD) analysis, 56
"Quality With a Name," 20
queuing theory. *See also* speed.
  average process time, calculating, 100–101
  cascading queues, 113–114

queuing theory *(continued)*:
cycle time reduction
establishing a cadence, 108–109
evening out work arrival, 103–105
limiting work to capacity, 110–111
minimizing process elements, 105–107
minimizing process size, 107–108
pull scheduling, 112–114
examples
airport check-in desk, 110
asynchronous cadence, 109
customer service, 111–112
doctor's appointments, 104–105
IT workload, 103–104
pull scheduling, 112–113
release cycles, 107–108
a seven year list, 106–107
thrashing, 111–112
Google, 101–102
Little's Law, 100–101
system stability, 101–102
utilization, 101–102
variation, 101–102
QuickBooks, 70
Quicken Rental Property Management, 57–58

R

raises as incentives, 144
Rally Software Development, 149–152
ranking people, 128, 142–143
Raymond, Eric, 54
red-eye reduction example, 162–163
refactoring, 164–168
Reichheld, Fred, 241
relational contracts, 219–221
relearning, 76
release cycles, example, 107–108
release planning, 179–181
remote teams, 212–213
repeatable reliable cycle time, 238
requirements
churn, 24, 91
nonfunctional, 182, 201
overloading, 25
SRS (Software Requirements Specifications), 75
stale, 74
test specs, 82
timing assumptions, 233
too early, 24, 91
respect for people, 3, 36–38
response time
by category, 84
at peak capacity, 101
queue length, 172
reliability, 98
testing, 201
responsibility, 56–57
responsibility-based planning and control, 133–135
retrospectives, 236
return on investment (ROI), 41, 240–241
reversible decisions, 32
rewards. *See also* compensation; incentives.
basis for, 144–145
intrinsic, 146
right to think, 237
risk
contracting away, 218
custom software development, 181
partially done work, 24
refactoring, 164
river companies, 141
robust development processes, 177
rocks-and-stream metaphor, 7–8
Rogers, Paul, 57
root causes
failure of lean initiatives, 153
group improvement failure, 174
low quality and productivity, 121, 123–124
of problems, 121, 123–124
technical debt, 150
waste, 67
rotating people, 212

S

safety considerations, stop-the-line culture, 9
sales, engineering, development (SED) system, 55
Sapolsky, Harvey, 179
satisfaction, customer, 49–52, 241
schedules
inventory. *See* Just-in-Time.
Kanban, 10–11
PatientKeeper releases, 97
philosophy of, 228
slipping dates, 133–134
and teams, 134, 135
Schnaith, Kent, 192
Schwaber, Ken, xvii
scientific method
Deming Cycle, 121
Just-in-Time commitment, 154
managing people, 121
problem solving, 154, 169–172
steps of, 154

stop-the-line culture, 154
Toyoda, Kiichiro, 154
Toyoda, Sakichi, 154
Toyota Production System, 154
scope bloat, 25
scope control, 25
Scrum
bottleneck elimination, xix
creation of, xvii–xviii
definition, 28
quality improvement, 28
stress avoidance, xix
Type A, xvii
Type B, xvii
Type C, xvii
waste elimination, xix
winning companies, xvii
winning product portfolio, xvii
winning teams, xvii
Scrum Product Owners, 133
ScrumMasters, 133
Sears, 34
SED (sales, engineering, development) system, 55
seiketsu (standardize), 191–192
seiri (sort), 191–192
seiso (shine), 191–192
seiton (systematize), 191–192
self-directing work, 137–141
self-organization, 17, 19, 97
set-based design, 160–164
seven principles
building quality in, 25–29
create knowledge, 29–32
defer commitment, 32–33
deliver fast, 34–35
eliminate waste, 23–25
myths
haste makes waste, 35
one best way, 37–38
optimize by decomposition, 40–41
planning is commitment, 33
predictable outcomes, 31–32
specifications reduce waste, 24–25
testing to find defects, 28–29
optimize the whole, 38– 41
respect people, 36–38
seven wastes. *See also* waste.
defects, 81–82
delays, 80–81
extra features, 75
handoffs, 77–78
partially done work, 74–75
relearning, 76
task switching, 78–80
seven year list, example, 106–107
shared leadership, 56
sharing early and often, 118
Shewhart Cycle, 121
Shimmings, Ian, 41
shine (seiso), 191–192
Shingo, Shigeo
introduction, 6–7
purpose of inspections, 82
seven wastes, 73
types of inspections, 27
ship builders, training, 234–236
shitsuke (sustain), 191–192
Shook, Jim, 35
Shore, Jim, 20
Sienna minivan, 53–55
Silicon Valley Product Group, 53
silos, 40, 131
simulation, kanban cards, 10
single point of responsibility, 65
Six Sigma, 124, 229–230
slack, 15, 88, 102, 112, 134
slipping dates, 133–134
Sloan, Alfred P., 2, 40–41
slogans as incentives, 123
small batches, 15, 74, 101–102, 196
Smalley, Art, 153
Smith, Levering, 178
Sobek, Durwood, 53
software
cost of maintenance, 20–21
development timeline, 20
difficult to change. *See* legacy systems.
embedded, definition, 20
enterprise, definition, 20
legacy, 166–168
structure of. *See* architecture, software.
software companies *versus* internal IT, 62–65
software development
capable processes, 98
concurrent, 182
defect queues, 25–26
detailed design, 29–30
deterministic school, 21
empirical school, 21
handling changes. *See* change, management.
large-batch approach, 71, 102
outsourcing, 216–217
plan-driven methods, 33

software development *(continued)*:
principles of. *See* principles; seven principles.
process quality measurement, 99
speed, competitive advantage, 35
speed *versus* hacking, 35
systematic learning, 31
waterfall model, 22, 29–30
software development, custom systems
accountability, 64–65
beginning/end criteria, 62
change requests, 62
funding profiles, 61
IT departments
accountability, 64–65
fixing, 64
versus software companies, 62–65
we-they model, 63
IT—business collaboration, 62–65
products *versus* projects, 60–63
software companies *versus* internal IT, 62–65
staffing, 62
we-they model, 63
Software Requirements Specifications (SRS), 75
sort (seiri), 191–192
Southwest Airlines, 11–12
span of influence *versus* span of control, 144–145
specialists in teams, 130–131
specification-by-example, 200
specifications, 24–25, 150
speed. *See also* deliver fast; queuing theory.
average projects, 99
cycle time, 98–99
dividing work into stories, 99
expediting, 98
versus hacking, 35
measuring, 99
PatientKeeper delivery cycle, 95–98
principle of, 45
process availability, 98
process capability, 98
red flags, 98
setting upper limits, 99
time delays, 98–99
unique projects, 100
Spolsky, Joel, 36
Spring, 150
Sprints, at PatientKeeper, xvii
SRS (Software Requirements Specifications), 75
staffing. *See* partners; people; teams.
Stalk, George, 5, 35
standardization, problem solving, 172
standardize (seiketsu), 191–192
standards for software development, 193–196
statistical process control, 120–122
stealing hangers, 125
stop-the-line culture
andon, 139–140
definition, 5–6
safety considerations, 9
and scientific method, 154
stories
dividing work into, 99
iterative development, 183–186
no partial credit, 188
story tests, 200. *See also* acceptance tests.
story-test driven development, 186
strangling legacy code, 167
stress (Mura), xix
stress avoidance, xix
Strong Project Leader, 54
suggestion systems, 236
supervisors. *See* people, managing.
suppliers, choosing, 122, 123
supply chain, lean, 12–13
sustain (shitsuke), 191–192
Sutcliffe, Kathleen M., 9
Sutherland, Jeff, 71, 96
synchronization, nested, 203–204
synergy, 121, 207–217
System of Profound Knowledge, 121
system stability and queuing theory, 101–102
system variation, 121
systematic learning, 31
systematize (seiton), 191–192
systems design stage, 47

T

T5 Agreement, 217–218
tacit knowledge, 14, 31, 77–78, 156–157
Takeuchi, Hirotaka, 156
target costs, 180, 218–219, 221
targets as incentives, 123
task switching, 78–80
Taxonomy of Problem Management Activities, 20
Taylor, Frederick Winslow, 2, 37, 227
TDD (test-driven development). *See* test-driven development.
teachers. *See* training.
teams. *See also* partners.
barriers to, 128
champions, 133
characteristics of, 126–127

chartering, 241
coaches, 133
co-located, 211, 213
complete, 57–60
cross-functional, 56, 64, 78, 122
dependencies, 135
design/build, 118, 123, 133
development
3M, 56–60
capacity, 99
champions, 132
DFSS (Design for Six Sigma), 229
error prevention, 82
expertise, 129–130, 212
goal of, 240
incentives, 123
interaction designers, 189
joined at the hip, 55
maintenance duties, 79
measurements, 237
pride in workmanship, 210
process improvement, 31
pull scheduling, 112–114
rewards, 145
set-based concurrent engineering, 16
size, and technical debt, 153
expertise, 129–131
global, 212
Honda, 55
versus individual efforts, 126
Intuit, 55, 57–58
leadership, 55, 132–133
limiting work to capacity, 134
organizational boundaries, 214
product managers, 133
Quicken, 55
ranking systems, 128
remote, 212–213
responsibility-based planning and control, 133–135
schedules, 134, 135
Scrum Product Owners, 133
ScrumMasters, 133
silos, 131
slipping dates, 133–134
specialists, 130–131
variation, 135
winning, xvii
work breakdown structure, 135
versus workgroups, 126–127, 212
Teamwork is the key..., 56–57
technical debt, 150
technical leadership, 132–133
technical success, 145
technical writers, 75, 130–131
test early, fail fast, 118–119
test harness
acceptance tests, 202
benefits of, 82
legacy systems, 166–167
schedule, 27
unit tests, 200
usability tests, 201
user interface, 151
test-and-fix churn, 24
test-driven development (TDD)
exploratory tests, 201
productivity, 28
property tests, 201
purpose of, 199
story tests, 200
types of tests, 199
unit tests, 200
usability tests, 201
testing
3D modeling, 118
automating, 82
Boeing 777, 118–120
business intent, 200
design intent, 200
to find defects, 28–29. *See also* test-driven development.
nonfunctional requirements, 201
testing early, failing fast, 118–119
too late, 88, 91
user interface, 150–151, 201
verification, role of, 29
testing early, failing fast, 118–119
tests
acceptance, 150, 186
acceptance-test-driven development, 186
exchanging, 212
programmer. *See* unit tests.
story-test driven development, 186
unit, 200
usability, 21
textile industry, Japan, 3–4
Theory of Constraints, 230–233
theory of knowledge, 121
thinking, 236–237
thinking tools, 21–22, 195
thrashing, example, 111–112
time, competing on the basis of, 34
timebox, 32, 181

timelines. *See* value streams.
too many things in processes, 105–107
too much work. *See* limiting work to capacity.
tools *versus* results, 229–230
towering technical competence...., 129
Toyoda, Eiji, 5, 226
Toyoda, Kiichiro
  incentives, 141
  introduction, 4
  scientific method, 154
  tracking knowledge, 155
Toyoda, Sakichi
  evolutionary thinking, 226–227
  incentives, 141
  introduction, 3
  scientific method, 154
  tracking knowledge, 155
Toyota
  chief engineer, 53–55
  competitive advantage, 224
  fire at Aisin plant, 208–209, 211
  genchi-genbutsu (go, see, confirm), 54
  versus other vehicle manufacturers, 13
  outsourcing, 216–217
  problem definition, 152–153
  product delivery deadlines, 161
  profits, xxiii
  responsibility, 56–57
  responsibility-based planning and control, 133–135
  set-based design, 161
  Sienna minivan, 53–55
  Smart Car initiative, 224–225
  Teamwork is the key..., 56–57
  towering technical competence...., 129
  training new engineers, 129
Toyota Product Development System
  See also Just-in-Time manufacturing
  See also mass production
  See also Toyota Production System
  cornerstone elements, 16
  entrepreneurial leadership, 16
  expert engineering workforce, 16
  respect for people, 36–37
  responsibility-based planning and control, 16
  set-based concurrent engineering, 16
  software development philosophy, 21
  study of, 15
Toyota Production System
  See also Just-in-Time manufacturing
  See also mass production
  See also Toyota Product Development System
  automated looms, 3
  autonomation (Jidoka), 5–6
  detecting abnormalities. *See* autonomation (Jidoka); stop-the-line culture.
  goals, 152–153
  Japanese auto industry, 4–7
  Just-in-Time flow, 4–5
  overview, 4–7
  push *versus* pull systems, 236–237
  scientific method, 154
  versus Six Sigma, 229–230
  thinking, 236–237
  value streams, 83
Toyota Production System, 5
The Toyota Way, 14
traceability, 75, 199
tracking knowledge, 155–159
tradeoffs, 41, 158, 241
training
  Allen's steps, 234–236
  Deming's points, 122, 123
  on the job, 234–236
  Job Instruction (JI) module, 235–236
  Job Methods (JM) module, 235–236
  Job Relations (JR) module, 235–236
  new engineers, 129
  ship builders, 234–236
  TWI (Training Within Industry), 235–236
  vocational education, 234–236
Training Within Industry (TWI), 235–236
transactions, outsourcing, 215
traveling team leaders, 213
trust, 125
Turner, Richard, 33
Type A Scrum, xvii
Type B Scrum, xvii
Type C Scrum, xvii

U

The Ultimate Question, 241
uncoded documentation, waste, 74
undeployed code, waste, 75
understand-before-doing, 19
undocumented code, waste, 75
unit tests, 200
United Airlines, 117–118
United Kingdom, 41, 193, 217
United States
  3M tour, 213
  Deming and, 121
  doctor's appointments, 104
  invention of interchangeable parts, 1–3

liens registry, 231
Toyota manufacturing, 216, 226
Toyota moves to, 12
unsynchronized code, waste, 74
untested code, waste, 74
unused documentation, waste, 77
US War Production Board, 235
usability tests, 201
used car sales, 41
user interface
competitive advantage, 189
iterative design, 189–190
testing, 150–151, 201
variation, 189–190
utilization
and cycle time, 102, 244
full, 88
Google workforce, 101–102
and queuing theory, 101–102
and variation, 101–114

V

value
customer-focused organizations
champions, 52–57
chief engineer, 53–55
complete teams, 57–60
decision making, 57
designing for manufacturability, 58–59
designing for operations, 58–59
development goal, 55
facilitating information flow, 52–60
leadership, 52–57
leadership teams, 55
Murphy's Law, 59–60
responsibility, 56–57
shared leadership, 56
What can go wrong, will go wrong, 59–60
customers
delighting, 49–52. *See also* Google.
focus on the job, 51–52
Kano model, 49–52
needs, 43
satisfaction, 49–52
understanding, 50
value principle, 44
value streams
churn, 91
delays, 91
examples, 85–91
for future processes, 92
keeping it simple, 85
mapping, 83–84
owner identification, 84–85
preparation, 83–84
start/stop points, 84
waste diagnosis, 91
Van Schooenderwoert, Nancy, 27
variation
inherent in the system, 121
and queuing theory, 101–102
and utilization, 101–114
variation in teams, 135
verification, and long release cycles, 107–108
verifying results of problem solving, 172
video cassettes, manufacturing, 59
visible signals, 139–140
vision, 16
visual workspace, 136–141
"vital few and trivial many" rule, 26
vocational education, 234–236
voice of the customer, 53, 229
volunteers, 54, 208–210

W

waiting. *See* delays.
Wake, Bill, 165
wall charts, 140
war room, 213
waste. *See also* seven wastes.
80/20 rule, 25–26
anticipating, 76
biggest source of, 24–25
churn, 24
complexity and, 67, 69–73
diagnosing. *See* value streams.
elimination
principle of, 23–25
reducing by specification, 24–25
Taiichi Ohno on, 23–25
extra features, 24–25
inventory as, 24
Just-in-Time commitment, 164
lost knowledge, 76
Muda, xix
multitasking, 78–80
non-value-added, 23, 83
partially done software, 24
recognizing, 23. *See also* value streams.
requirements churn, 24
root cause, 67
test-and-fix churn, 24
uncoded documentation, 74
undeployed code, 75

waste *(continued)*:
undocumented code, 75
unsynchronized code, 74
untested code, 74
unused documentation, 77
"vital few and trivial many" rule, 26
waste (Muda), xix
waterfall development model, 22, 29–30
Weick, Karl E., 9
Welch, Jack, 173
we-they model, 63
What can go wrong, will go wrong, 59–60
"When IT's Customers Are External," 62–63
Whitney, Eli, 1
Who has the D?, 57
Wild, Werner, 159
winning companies, xvii
winning product portfolio, xvii
winning teams, xvii
Wolf, Bob, 208
Womack, James, 43
work breakdown structure, 135
workers. *See* partners; people; teams.
workgroups, 126–127, 212
Working Effectively With Legacy Code, 167
Working Together program, 118–120
Workout, 173–175
write less code, 29, 67–73

Y

YAGNI (You Aren't Going to Need It), 165
Yamada, Kosaku, 13
Yokoya, Yuji, 53–55

Z

Zara, 67–68
zero inspection, 6–7