INDEX

10%-of-the-project rule of thumb, 118
80/20 rule, 337

A
Ability to make informed decision, 110
Acceptance test releases, incomplete and unrealistic, 351–352
Acceptance test suites, 315
Acceptance testing, 10, 24, 519
Access permissions, 42
Accidental Project Manager, The (Ensworth), 223, 257, 452
Actionable estimates, 134–135
Activity, 70, 519
Adapting to change, 54–56
After the Gold Rush (McConnel), 242
Against the Gods (Bernstein), 57
Agile methodologies, 172
All-pairs, 295
Alpha testing, 10, 17, 24, 519
Amazon.com, 245
Ambiguities, separating from bugs and features, 408–410
AmiBug, 491
AmiBug, Inc. Web site, 71, 431
Amortize, 112, 519
Anna Karenina (Tolstoy), xxv
ANSI/ISO 9126 standard, 48–49
Application domain knowledge, 232–233
Application domain skills, 233, 519
Archimedes, 273
Art of Software Testing, The (Myers), 172, 311, 336, 431
Art of War (Sun Tsu), 456
“At Your Service,” 265, 486
ATEs (automated test engineers), 74
Attack on Big Build, 360-368
Attention to detail, 232
Auditability, 290
Audition interview, 519
Automated functional testing, 113
Automated functionality regression tests, updating, 89
Automated Software Testing (Dustin, Raskha, and Paul), 172–173, 179, 313
Automated Test Life-Cycle Methodology, 173
Automated test scripts, 112
Automated testing, 179, 311–312
Automation, 291

B
Bach, James, 199, 232, 409
Bad Software (Kaner), 5
Balanced scorecard, 422, 441, 519, 521
Balanced testing, 41, 42, 306
Bartlett, Bob, 101
*Basics of FMEA, The* (McDermott et al.), 38
Basis tests, 500
Becker, Randall, 326, 461
Behavioral coverage, 298
Behavioral coverage assessments, 299
Behavioral test coverage, 300
Behavioral testing, 24–26, 173
Behavioral tests, 25, 313, 520
Beizer, Boris, 268
Berger, Bernie, 232
*Best Practices for Software Testing* (Drabick), 492
Beta test sites, 282
Beta testing, 10, 17–18, 24, 180, 519–520
Big Build
  attack on, 360–368
  big bug, 392–398
  test results, 422–440
Black-box coverage, 298, 304
Black-box testing, 173
*Black-Box Testing* (Beizer), 173, 280, 296, 298
Black-box tests, 25, 520
Bolton, Michael, 293
Bottom-up estimation, 81
Box, George, 20
Brainstorming
  quality risk analysis, 38
  quantifying risk, 38–42
  ranking failure, 38
Brooks, Fred, 145
Brook’s Law, 146
Budgets, 97, 419
  alliances for selling approach, 124
  amortizing test development work and configuration of test equipment, 115
  analyzing ROI (Return on Investment) for testing, 98–113
  benefits from testing plan, 118–119
  compromising on, 128–129
  contingency, 128
  cost-of-quality equation, 117
  cutting, 126–127
  external failure, 116–117
  Failure Mode and Effect Analysis worksheet, 127
  found and not fixed bugs, 117
  fully burdened staff rate, 115–116
  hardware, 113
  internal failure for test team, 116
  minimum investment in testing, 123
  mitigating risks, 127
  must-fix bugs, 116–117
  project tracking information, 118
  project-based team assignment strategy, 113
  providing services and products customers need, 124
  quality risk analysis process, 117
  resources, 113–114
  ROI (return on investment), 115
  saving money in, 128–129
  software, 113
  Sumatra Project, 113–119
  test coverage analysis, 127
  testing tools, 113
  usability testing, 113
  value of risk mitigation, 117–118
  work-breakdown-structure, 127
  working with other departments, 127
Bug crawls, 444
Bug fixes, 351
Bug report ping-pong, 404
Bug reports, 346, 387–388, 520
  anger discussing, 414
  bias, 375
  bug assignment, 405
  bug workflow, 417
  bugs fixed by accident, 410–412
  bugs that aren’t bugs, 412
  building trust and credibility with programmers, 413–414
  challenges, 408–415
  clear boundary between testing and debugging, 406–408
  commercial bug tracking tool, 403–404
communications channels, 412–413
condensing, 390, 395–396
defect metrics, 404
digital pictures, 403
effectively communicating useful information, 402–405
ensuring quality, 391
error logs, 403
escalating problems with, 412–413
explaining expected behavior, 410
flexibility, 414
improvements, 416–418
improving quality, 414
internal tracing information, 403
irreproducible symptoms, 410–412
linking results to test cases, 400
making individuals look bad, 413
memory contents, 403
metrics, 418
neutralizing wording, 391
one report per symptom, 405–406
open-mindedness about, 414
peer reviews, 397, 417–418
priorities, 417
procedures, 416
process, 388–391, 402–408
process improvement efforts, 416–417
programmers as key internal customers, 409
project managers as key internal customers, 409
project-wide, service-oriented mind set, 414
removing confusing or misleading words, 390–391
rephrasing text, 396–397
reviewing, 391
screen captures, 403
selecting bug tracking tools, 414–415
separating bugs, features, and ambiguities, 408–410
separation of duties, 407
software version, 399
steps to reproduce bug, 392, 394
structure, 389
summarizing, 390, 394
system development process, 412
system under test, 412
test scripts, 416
tester attitudes, 416
testing history, 402
workaround information, 404
working files, 403
Bug tracking database, programmer notes in, 412
Bug tracking tools
adapting to bug reports, 417
obtaining, 417
selecting, 414–415
Bug triage, 206
Bug triage committee, 370, 408
Bug triage processes, 390
Bugs, 8, 520
assignment, 405
average costs per, 101
beyond immediate testing objectives, 369–370
Big Build, 392–398
capturing affected subsystem, component, or function, 400
connecting to real usage, 409
cosmetic, 40, 398
defect removal models, 401
desirable, 399
discretionary, 399
essential, 399
finding and fixing earlier, 371
finding before testing, 52–53
fixed by accident, 410–412
general case, 393
generalizing, 389–390
impeding detection, 53
intermittent, 375, 389, 411–412
irreproducible, 353, 410–412
isolating, 389, 393–394
lifecycle, 414
loss of data, 39, 398
loss of functionality, 39, 398
Bugs, continued
loss of functionality with workaround, 40, 398
mistakes in system configuration, 117
one bug report per symptom, 405–406
partial loss of functionality, 40, 398
predicting amount of time for, 383
priorities, 399
reporting, 41, 306
reproducing, 389
ROI (return on investment), 115–116
separating from features and ambiguities, 408–410
severity, 398–399
specific configuration tested, 400
steps to reproduce, 392, 394
test design preventing, 314
tester errors, 117
that aren’t bugs, 412
that don’t get fixed—but are known and ROI, 106–107
that get fixed or prevented and ROI, 99–106
tied to specific requirements, 400
unimportant, 411
urgent, 399
valuable, 399
whether to fix, 477
workflow, 417
Build ID, 342, 520
Build manager, 349, 520, 526
Build plan, 206, 520
Build release, 520
Builds, 327
information about code fix, 400–401
Built-in profiling tools, 305
Bullock, James, 122
Business risk, reducing, 178
Buwalda, Hans, 279

C
Calculating defect costs, 102–103
“Calculating the Value of Testing,” 122
Candidates, 220, 272, 520
Canned test plans, 207
Capability Maturity Model, The (Paulk et al.), 30, 490, 503
Capturing knowledge, 291
Career growth and skills assessment worksheet, 252
Career path, 248, 254
long-term, 257–262
rite of passage, 261
tester track, 261–262
testers, 272
Cause-and-effect analysis, 295
CCB (Change control board), 284, 344, 370, 408, 466
CCB (change control boards), 56
“Cem Kaner on Rethinking Software Metrics,” 449
Certification training, 242–243
Cetus-Links Web site, 20
Challenges, handling, xxxv
Change
adapting to, 54–56
complex nature of effects, 478–480
ripple effects of, 480
Change control, 390
Change management, 461
balancing features, schedule, budget and quality, 477–478
benefits and opportunities, 482
challenges, 478–482
complex nature of change effects, 478–480
costs and risks, 482
giving stakeholders a voice, 477–478
as interconnected process, 474–475
objectivity, 482
perceived as roadblock, 481–482
politics, 483
process, 462–464, 475–478
project timing, 464
quality risk analysis document, 55
right time to make changes, 482–483
ripple effects of change, 480
selecting right changes in right order, 476
Sumatra project, 464–474
test team participation, 474–475
Change management board, 344
Change request, source of, 462
Checklists, xxviii
CiteSeer Web site, 180
“Classic Testing Mistakes,” 369
CMM in Practice (Jalote), 94, 401, 441
Code, 17
Code and fix, 21, 520–521
Code churn, 351
Code complexity tool, 303
Code coverage, 300
Code coverage tool, 303
Code swapping, 25
COE (cost of exposure), 49, 60
Collaborative processes, xxix–xxx
Collard, Ross, 293, 326, 409, 461, 498
College degree, 242
Combinatorial explosions, 293
  all-pairs, 295
  cause-and-effect analysis, 295
  design of experiments, 295
  distributing test cases across different configurations, 294
  domain analysis, 295–296
  equivalence partitioning, 295–296
  hazard analysis, 295
  identifying important configurations, 294
  orthogonal arrays, 295
  paradigmatic uses or workflows through system, 296
  Taguchi methods, 295
  varying configurations running test against, 294
Commercial bug tracking tool, 403–404
Commitment, 197–199
Common expectations, 197–199
Companies, damage to reputation, 119
Complete Guide to Software Testing (Hetzel), 172, 300, 314
Component testing, 10, 17, 24, 25, 28, 521
Computer-Related Risks (Neumann), 48, 176, 291
Configurations, identifying important, 294
Confirmation testing, 369
Consensus, 197–199
Consultants, 267, 269–271
Consulting test professional, 27–28
Context
diagram, 206
operational, 11–14
organizational, 11–14
understanding, 10–11
Context-discovery process, 12–14
Continuity, 128
Continuing education, 241, 242
Contract Professional magazine, 263
Contractor test technicians, 237
Contractors, 269–271
Copeland, Lee, 199
Core dumps, 403
Cost of quality, 104, 106
Cost of quality analysis, 99–106
costs of conformance, 99
costs of nonconformance, 100
Cost of total failure, 109
Cost-of-nonconformance, 110
curve, 109
policy, 108–109
savings, 141
Cost-of-quality equation, 117
Costs, amortized across multiple projects, 112
Costs of conformance, 99, 104, 141
Costs of external failure, 100–101
Costs of internal failure, 100–101
Costs of nonconformance, 103
Costs of prevention, 100
Coutre, Robert, 66
Craft of Software Testing (Marick), 25
Craig, Rick, 387
Crazy Horse, 11
“Credible Estimation of Small Project,” 90
Critical network, 134, 521
Critical path, 134, 521
Critical processes, xxvi
Critical quality risk, 124
Critical risks and test system, 273
Critical skills
  application domain knowledge, 232–233
  attention to detail, 232
  general qualifications, 232
  skills profile, 234–235
  technical expertise, 232–233
  testing skills, 232–233
  top-down approach, 232
  verbal and written communication, 232
Critical skills spreadsheet, 253, 263–264
Critical testing processes, xxv–xxvii
  collaborative, xxix–xxx
  context, xxix, xxx, xxiii
  internal, xxix–xxx
Cultures, 381
CUPRIMO (capability, usability, performance, reliability, installability, maintainability, and operability), 49
Cursory testing, 42
Custer (Wert), 11
Custer, George Armstrong, 10–11
Customers, 4, 45, 521
Cyclomatic complexity, 299–300

D
Daich, Gregory, 293
Damage to company reputation, 119
Dashboard, 422, 521
Dawson, Gary, 326, 461
Death March (Yourdon), 456
Debug build, 403
Debugging, xxvi–xxvii
  boundary with testing, 406–408
  test environment, 184
Defect detection, 401
Defect detection percentage, 508
Defect injection, 401
Defect metrics, 404
Defect removal, 401
Defect removal models, 94, 401
Defects, 8, 102–103, 520, 521
Defining job accurately, 256–257
“Delivering Unwelcome News to Developers,” 457
Delphic Oracle, 90
DeMarco, Tom, 371
Deming, W. E., 6, 20
Deming Management Method, The
  (Walton), 6
DeNardis, Chris, 214
Department of Defense 2617A standard, 398
Dependencies, 23, 76, 134
Dependent tasks, 76
Design of experiments, 295
Design-based tests, 174
Deutsch, Harvey, 326, 461, 476
Development plan, 206, 522
Development teams and junior programmer roles, 28
Digital pictures, 403
DLL hell, 294
Document management subsystem, 36–37
Document management system edit engine, 279–280
Documentation
  ambiguity, 288–291
  amount, 288
  appropriate, 287–293, 321
  auditability, 291
  automated tests, 291
  automation, 291
  beliefs, rational and otherwise, 293
  capturing knowledge, 291
  complexity of test execution process, 291
  degree test system is product, 291
  efficiency, reuse, and maintenance, 292
  formalizing and standardizing, 291
  guiding inexperienced testers, 291
  lack of knowledge and chaos, 293
  motivational concerns, 292
  regression testing, 291
  reproducibility, 291
  risk level of system, 291–291
sharing knowledge, 291
skill levels of testers, 292
test and project stakeholder expectation and requirements, 292
test cases, 402
test managers, 449
test plan, 202–204
test releases, 344–347
tight time windows for test execution, 291
time and resources available to design and implement test cases, 292
unambiguous, 291
written test cases, 291
Domain analysis, 295–296
Don Quixote (Cervantes), 453
Drabick, Roger, 492
Draft/feedback/update/finalize cycle, 84
Dumb monkeys, 280, 281
Dyes, Tim, 326, 461
Dynamic tests, 313
Dyson, Esther, 66

Estimates
accepting context and constraints of project, 136
assigning responsibility for action, 134–135
bogus, 149
challenges, 137–150
completeness, 133
compromises, 151–152
excessive external constraints, 133
factors affecting, 129–132
failure, 148–149
funding problems, 133
imprecision of, 144–145
initial, 156
lack of agreement, 133
lack of knowledge, 133
mythical man-month, 145–146
new technologies and, 132
nine-month rule, 145–146
nonstakeholders, 147–148
obstacles, 151
overly optimistic, 156
people outside scope of project, 147
practicing, 151
project level, 146
quick, 137
rational task durations, 134
realistic picture of project’s future, 132–134
realistic schedule, 133–134
reopening debate about project schedule and budget, 123
reviewing other, 151
revising, 145
selling, 121–126
selling based on risks, 124
stretching goals, 133
studying topic, 150–151
team performance, 132
tester-to-programmer ratio-based, 82
unrealistic scheduling, 146
worked backward from arbitrary ship date, 146

E
Edit engine for document management system, 279–280
Education, 241–242
Effective Project Management (Wysocki et al.), 67, 90
Efficient, 307
Emergency patches, 340
Empirical approach, 180
Employee skills assessment, 252
Engineering plan, 520
Engineering Services group, 14
Entertainment systems risks, 306
Envisioning Information (Tufte), 443
Equivalence partitioning, 295–296
Error logs, 403
“Estimated Bug Find-Fix Time.xls” template, 94
Estimated schedule, developing, 80

Don Quixote (Cervantes), 453
Draft/feedback/update/finalize cycle, 84
Dumb monkeys, 280, 281
Dyes, Tim, 326, 461
Dynamic tests, 313
Dyson, Esther, 66
Estimating projects
management support, 121–126
obstacles, 121–126
planning phase, 83–86
rules of thumb, 81–82
staffing, 86
test development, 87–91
test environment configuration, 91–92
test execution, 92–95
Estimating risk, 65–66
process, 67–69
Sumatra Project, 70–80
Estimating Software Costs (Jones), 81, 90, 111
"Estimating Tester to Developer Ratios (or Not)," 82
Estimation, 66, 522
bottom-up and top-down, 81
Evening shift, 379
Evolutionary model, 346
Excessive risk aversion, 58
Executive summary, 170
Expectation of quality, 5, 522
Experience of quality, 5, 522
"Exploratory Planning," 200
"Exploratory Testing and the Planning Myth," 200
"Exploring, Discovering and Exterminating Bug Clusters in Web Applications," 431
Extended shifts, 379–381, 522
Extensive testing, 41, 306
External failure, 100–101, 116–117
Extreme Programming model, 23, 25, 172

F
Failure
likelihood, 40
ranking, 38–42
Failure mode, 36, 522–523
Failure Mode and Effect Analysis (Stamatis), 38, 50, 59
Failure Mode and Effect Analysis worksheet
budgets, 127
interface risks, 77
Recommended Action field, 76–77
updating, 84
verifying complete coverage of risks, 76
Family of systems, xxxiv
Faught, Danny, 300, 326, 461
Features, 419
separating from bugs and ambiguities, 408–410
Fiat certifications, 243
FMEA (Failure Mode and Effect Analysis), 37–38, 58, 83–84
access permissions, 42
COE (cost of exposure), 50, 60
defect detection for requirements document, 42, 44
good written project documents, 48
no written project documents, 48
priority and likelihood, 61
QFD (Quality Function Deployment) technique, 50
quantifying risk, 38
risk levels, 50
RPN (risk priority numbers), 60
severity of potential problem, 61
used for quality assurance, 53
Formality, 490–497
Found and not fixed bugs, 117
"Four Ways Testing Adds Value," 98
Fouts, Peggy, 326, 461
FRUEMP (functionality, reliability, usability, efficiency, maintainability, and portability), 48, 49
Fully burdened staff rate, 115, 116, 523

G
GA (General Availability) Build, 72, 523
General qualifications, 232
Glass-box testing, 173
Gold Master, 72, 523
Golden Build, 71, 72, 523
Golden Candidate, 72, 523
Golden Candidate test release, 92, 94
Goldsmith, Robin, 293
Graveyard shift, 379
Gray-box coverage, 304
Gray-box testing, 299
Guide to Quality Control (Ishikawa), 295
Guide to the Project Management Body of Knowledge, The, 66

H
Handling challenges, xxxv
Hardware
  budgets, 113
  usage plan, 156–157
Hazard analysis, 295
Hendrickson, Elisabeth, 232
Hierarchical storage management systems, 281
Hiring and critical skills spreadsheet, 264
  “Hiring Technical People,” 247
Hitchhiker’s Guide to the Galaxy, The (Adams), 125
HSM (hierarchical storage management) devices, 190–191
Human Resources Department, 221–222, 523
  “Human Side of Risk, The,” 57

I
I Am a Bug! (Sabourin), 476
Iberle, Kathy, 409, 498
IEEE (Institute of Electrical and Electronics Engineers) format, 202–203
Implementing the IEEE Software Engineering Standards (Schmidt), 493
Improvements, implementing, xxxv–xxxvi
Incremental development model, 14, 16
Incremental model, 18, 174
Incremental system development lifecycle, 21, 94, 523
Independent test team, 24, 524
  adequate share of resources, 26
  moving into development groups, 30
  reporting findings, 26
  “Influential Test Manager, The,” 447
Information
  communicating, 31–32
  generating, 31
  internal and external dissemination, 370–371
  managing inward, 31
  useful, credible, and timely, 441–444
Informed decision, ability to make, 110
Input-Process-Output diagrams, 492
Insurance, 108–109
Integration and system test, 24
Integration and System Test team, 14
Integration plan, 206, 524
Integration testing, 10, 17, 72–73, 302–303
  arrival of certain components, 95
  continuation criteria, 163
  entry criteria, 163
  exit criteria, 163
  product testing, 524
  staffing needs and pass durations, 74–76
Integration testing backbones, 95
Integration testing plan, 206
Intermittent bugs, 375
Internal failure
  costs of, 100–101
  test team, 116
Internal processes, xxix–xxx
Internet time, 119
Interviews for candidates, 272
Introduction to the Personal Software Process (Humphrey), 94, 314
Irreproducible bugs, 353
IS/IS NOT table, 155
ISO 9000 for Software Developers (Schmauch), 490
Isolation, 393–394
  “It Depends: Deciding on the Correct Ratio of Developers to Testers,” 81
J
J. Rothman Web site, 82, 223, 247, 319, 456, 513
James Bach Web site, 295
Jefferson, Thomas, 202
Job descriptions, 224, 256–257, 259–260, 263
Jones, Capers, 81
Jorgensen, Alan, 409
*Journal of Software Testing Professionals*, xix
Juran, J. M., 337, 502
*Juran on Planning for Quality* (Juran), 4

K
K. Iberle Web site, 82
Kaner, Cem, 409
Kaner Web site, 5, 144, 313, 449
Kelvin, Lord, 60
Key players
compromising with, 210
leveraging relationships with other managers, 212
not stakeholders in project, 212
not supporting test plans, 209–213
perception of reasonableness of position, 212–213
role to play, 210
Key quality and testing stakeholders, 36, 524
Key stakeholders, 36, 133, 524
King Hieron, 273
Koomen, Tim, 81, 355, 383
Kruchten, Philippe, 512

L
Lawrence, Brian, 419
*Leadership Through Quality*, 509
*Lessons Learned in Software Testing*
(Kaner, Bach, and Pettichord), 172, 313, 369, 420, 462
Library, 327, 524
Lifecycles
processes, 16
testing within, 18–23
Lifecycles models
accuracy, 20–21
adaptations, 21
code-and-fix frenzy, 23
dependencies, 23
design and implementation of test system, 23
flexibility fitting testing within, 23
object-oriented development, 19–20
selecting best, 19
sequencing, 23
testers involvement, 23
Limited Availability, 72
Lind, Bill, 207
Lint, 24
Live testing, 24, 174, 180
Load testing, 280
LogiGear Corporation Web site, 232
Long-term career path, 257–262
Low-level employees, 222
Lumsoft Web site, 295

M
MacNary, Reynolds, 453, 507
*Madame Bovary* (Flaubert), 204
Maintenance teams and junior programmer roles, 28
Management
bogus estimates, 149
dimensions, 31–33
inward, 31
outward, 32–33
support for estimate, 121–126
upward, 32–33
“Manager Heal Thyself,” 513
Managers, 123
aware of value testing, 123
failure to communicate with, 122
importance of testing, 122
insight into quality, 110
interacting directly with, 31
limited time to convince, 123
little experience with well-run testing project, 123
outsourcing, 208
pay scale for testers, 263
seeing challenges in testing, 125
testing as big black hole, 98
testing as value-adding activity, 98
who don’t understand testing, 123
Managing
  inward, 31
 *Managing the Testing Process*, xix, xvii, xxvii, 179, 378
---
Manleitner, Markus, 348
Marick, Brian, 419
Martin, Darcy, 326, 461
Master test plan, 154–155
*Mastering Software Test Design* (Copeland), 200, 280, 295, 296, 300
McCabe, Thomas, 299–300
McCandless, Deborah, 208, 247
"Meaningful Metrics," 441
"Measured Response, A," 449
"Measuring Software Product Quality during Testing," 49
Memory contents, 403
Military
  education, 241
  organizations, 222–223
"Mission Made Possible" (Black and Kubaczkowski), 17, 328
Mitigating risks, 127
Morale, 26
"More Testing, Worse Quality," 508
MTE (manual test engineer), 74
Must-fix bugs, 116–117
Mythical man-month, 145–146
*Mythical Man-Month, The* (Brooks), 123, 146, 151, 336, 371, 376, 431, 453
*Mythical Man-Month, The, Second Edition* (Brooks), xxii
---
N
Need-to-know information, 266–267
Net present value of money, 141, 144
New hires, 222, 524
New technologies and estimates, 132
Newton, 119
Nguyen, Hung, 232
Nine-month rule, 145–146
No testing, 42
Nonbugs, 117
Nonstakeholders and estimates, 147–148
O
"Object-Oriented Programs and Testing," 180
Obtain Hiring/Contract Reqs task, 86
Offer letter, 222, 231, 524
Operational context, 11–14
Opportunity testing, 41–42, 306
Organizational context, 11–14
Organizations
  lifecycle processes, 13, 16
  quality-related investments, 3–5
  risks, 3
  systems, 3
Orthogonal arrays, 295
"Orthogonally Speaking," 295
*Out of the Crisis* (Deming), 63
Outsourced testing, 379–381
Outsourcing plan, 208–209
Outward management, 32–33
Overtime, 379
---
P
P. Hadke Associates Web site, 295
Palm Pilot, 119
Pareto Principle, 337
Pascal, Blaise, 390
*Patterns in Software System Failure and Success*, 111
PDA (Personal Digital Assistant) market, 119
Peer-level stakeholders, 18
Pemmaraju, Kamesh, 498
Peopleware (DeMarco and Lister), 377, 456
Perfecting, 6
Performance, Load, Capacity, and Volume test suite, 88, 275, 560
Performance testing, 25, 174, 182
Performing, 6
Permanent test technicians, 237
Per-release but-find-rate metrics, 342
Personnel Department, 222, 523, 524
“Perspectives from a Test Manager,” 231
Pervasive testing, 487–488
Pettichord, Bret, 409
Pettichord, Brit, 389
Phase, 70, 524
Pilot testing, 10, 524
pkgadd utility, 341
pkgrm utility, 341
Plan-Do-Check-Act process, 6
Planning, 6
Planning phase
analyzing quality risks, 83–84
draft/feedback/update/finalize cycle, 84
estimated schedule and budget, 85
estimation and planning activities, 84–85
management support and concurrence on finalized estimate, 85
one-on-one interview approach, 84
stable quality risk analysis, 84
test plan, 85
understanding project context, 83
updating Failure Mode and Effect Analysis document, 84
PMI Web site, 66
Pol, Martin, 81
Ponce de Leon, Juan, xxii
Preparing, 6
Principles of Quality Costs (Campanella et al.), 100, 104
Principles of Scientific Management, The (Taylor), xxii
Prioritization process, 206
Procedures and bug reports, 416
Processes, xxv–xxvi, xxvii
as checklists, xxviii
critical, xxvi
recognizing good, xxxiv–xxxv
simplicity, 341
Products
balance between quality, schedule, budget, and features, 27
risks, 81
road map, 45
testing, 10
Professional pessimism, 244
Professional Tester magazine, xix, 205
Programmers
building trust and credibility with, 413–414
as key internal customers for bug reports, 409
testers or test managers pressuring, 413
Project management teams
excessive external constraints, 133
lack of knowledge, 133
stretching goals, 133
Project managers
as key internal customers for bug reports, 409
rewarded based on profitability of project, 141
Project work-breakdown-structure, 134
Project-based test team, 265
Projects, 66, 525
behavioral testing, 24–25
budgets, 419
context, 83
contributor versus testers, 12
features, 419
flawed schedule, 146
lacking predictability and regularity of test releases, 338
live testing, 24
process risks, 81
quality, 419
quality risks, 58–59
readiness, 194–195
realistic picture of future, 132–134
reducing failure with better tracking practices, 111
reducing risk of failure, 12
relating test status to progress on, 444–447
schedules, 419
structural testing, 24–25
success of, 137
team information dissemination, 370–371
tracking information budget, 118
unrealistic scheduling, 146
Proxy stakeholders and quality risk analysis, 45, 48
Psychology of Computer Programming, The (Weinberg), 173

Q
Quality, 4, 119, 273, 419, 525
assessing, 273
capable tools for assessment, 305–307
endangering, 5
expectation of, 5
experience of, 5
importance of, 122
risks, 305–307
showing trends and destinations over time, 447–448
Quality assurance, 13, 104, 505
contrasting with testing, 525
Quality control, inadequate, 125
Quality Control Handbook (Juran and Gryna), 100
Quality Control Handbook, Fourth Edition (Juran and Gryna), 104
“Quality Cost Analysis: Benefits and Risks,” 144
Quality Functional Deployment, 76, 77
Quality Is Free (Crosby), 4, 100
“Quality Meets the CEO,” 126
Quality Professional magazine, 263
Quality risk analysis, 124
adapting to change, 54–56
ANSI/ISO 9126 standard, 48–49
brainstorming, 38
changing techniques during, 50
COE (cost of exposure), 49
customers, 45
degree of risk, 305–306
design specification and, 54
differentiating between types of risks, 50–54
estimate failure, 148–149
finding potential bugs before testing, 52–53
FMEA (Failure Mode and Effect Analysis), 37–38, 49, 50
handling challenges, 56–63
identifying key stakeholders, 63
implementing improvements, 63–64
informal technique, 50
informal techniques, 48, 49
initiating and facilitating, 59–60
instituting, 63
likelihood of failure, 40
load, capacity, and volume risks, 278
pointing toward solutions, 53
prioritized list of risks, 50
prioritizing risk, 40
properly framing discussion on, 52
proxy stakeholders, 45, 48
quality risks from various sources, 58–59
quantification of risk, 60–61
rationally dealing with risk, 56–58
reducing number of bugs, 53
reductions in scope of testing, 97
researching appropriate categories, 63
risk officer, 59
RPN (risk priority number), 41, 42
safety- or mission-critical systems, 149–150
severity of risk, 39–40
stable, 84
stakeholders existing but not available, 45
techniques for analysis, 48–50
test execution, 368–369
users, 45
variation in ratings, 61–62
Quality risk analysis, continued
  when to start, 54
  willingness of stakeholders to participate, 62–63
Quality risk analysis document, 55
  regular updates, 56
Quality risk analysis process, 36–44
  budget, 117
  involving correct participants, 44–48
Quality risk coverage, 298
Quality risks, 36, 58–59, 73, 110, 208, 275, 277, 525
  analyzing, 83–84
  balanced testing, 306
  balancing, 196
  bug fixes creating new problems, 369
  bug fixes that don’t resolve problem, 369
  critical, 124
  extensive testing, 306
  identifying and prioritizing, 37–38
  opportunity testing, 306
  reporting observed bugs, 306
  test strategies, 155
Quality Tree Web site, 508
Quantification of risk, 60–61

R
  Rashka, Jeff, 81
Rational Software, 512
Rational Unified Process, 19
readme files, 346
Realistic picture of project’s future, 132–134
Realistic schedule and estimates, 133–134
Reasonable user expectations, 286
“Recruiting Software Testers,” 247
Reference platform, 147
Reference system, 286, 526
Regression gap, 128–129
Regression risk, 176
  advanced object-oriented programming for limiting, 180
  management strategies, 176–180
  reducing, 179
  repeating tests, 369
Regression test gap, 177, 526
Regression testing, 177, 369, 526
  documentation, 291
  partial rather than full, 179
  reducing risk of allowing unrealistic test releases, 352
Regressions, 526
  affecting customer’s experience of quality, 176
  analytical approach, 179–180
  bet testing, 180
  code coverage measurements, 180
  extensive live testing, 180
  found by customer or user, 176
  increasing cost of internal failure, 177
  incurred cost of external failure, 177
  new features, 177–178
  offending change, 176
  previous release, 177–178
  risk, 177
  scenarios and foreseeable outcomes, 178
  undesirable complications and delays into debugging process, 176
Release, 327, 526
Release and smoke test, 332
“Release Criteria,” 456
Release engineer, 348, 349, 526
Release Engineering Plan, 206, 526
Release manager, 349, 526
Release name, 342, 520, 526
Release notes, 346
Reliability/Stability suite, 360
Repetative tests, 179
Repetitive tasks, 235–236
Reporting observed bugs, 41, 306
Repository, 327
Repository library, 526
Reproducibility, 291
Requirements, 317–318
Requirements document and defect detection, 42, 44
Resource assignments, 76
Resources
  dependencies, 134
  mismanagement, 377
  test environment, 181–183
Resumes, 224, 229–231, 248
Retaining test team, 263–264
Retesting third-party components, 316–317
Revision, 342, 520, 527
Revision number, 342, 520, 527
Rex Black Consulting Web site, 57, 66, 98, 328
  Publications page, 181
Risk analysis, 302
Risk assessment and allocating resources, 64
“Risk in Risk Management, The” (de Jaeger), 57
Risk management, 208
Risk officer, 59
Risks, 5
  automated tests, 312
  changes to, 55
  coverage of, 308
  dealing rationally with, 56–58
  degree of, 305–306
  differentiating types, 50–52
  entertainment systems, 306
  estimating, 65–66
  excessive aversion to, 58
  irrational factors, 57
  misleading aspect, 57
  mitigating, 41, 127
  predictable, 332
  preventing, 53
  prioritizing, 40, 50
  product, 81
  project and process, 81
  quality, 58–59, 305–307
  quantifying, 38–42, 60–61
  relative importance, 60
  ROI and tests mitigating, 107–110
  RPN (risk priority number), 42
  safety-critical systems, 306
  severity, 59–40
  test automation, 312
  unpredictable, 352
  variation in ratings, 61–62
  Rite of passage, 261
  ROI (return on investment), 98, 526–527
  accounting and incentive practices
    working against, 138, 141
  budgets, 115
  bugs, 115–116
  bugs that don't get fixed—but are known, 106–107
  bugs that get fixed or prevented, 99–106
  business case for testing, 112–113
  connecting with why managers invest in testing, 125
  guiding project, 110–112
  net present value of money, 141, 144
  positive, 123
  significance of numbers, 124–125
  testing, 98–113
  tests that mitigate risks, 107–110
  Role as testing professional, 122
  Rothman, Johanna, 101, 144, 358, 391
  Rough estimations and work-breakdown-structure, 81–82
  RPN (risk priority numbers), 41, 42, 60
  Ruf, Barbara, 326, 461
  Rules of thumb, 81–82, 90–91
  S
  Sabourin, Robert, 71, 239, 268, 491–492
  Safety-critical systems, 291–291
    quality risk analysis, 149–150
    risks, 306
  Salaries, 262–263
  Salary.com Web site, 263
  Sanity test, 315, 527
  Satisfice Web site, 295, 313, 369
  Schedules, 419
    minimum investment in testing, 123
    worked backward from arbitrary ship date, 146
  Scratch files, 403
Screen captures, 403
Screen shots, 403
Scripted manual tests, 146
SD Magazine Web site, 247
Secrets of Consulting, The (Weinberg), 12
Security testing, 25
Selling testing, 124
Sequencing, 23
Service-based test team, 27–28, 265
Shelfware, 308
Sitting Bull, 11
Skills assessment of employees, 252
Skills assessment worksheet, 249–250, 252
Skills growth, 249–252
Skills profile, 234–235
Skills review, 253
Skills-based test team, 265
SME (subject matter expert), 27
Smoke test, 315, 353, 527
Smoke test suites, 315
Soap opera approach, 279
Soap opera scenario, 280
SoftTest Web site, 490, 492
Software and budgets, 113
Software Cafeteria, 14
“Software Configuration Management for Project Leaders,” 185
Software Dioxide, xix
Software Engineering, Fourth Edition (Pressman), 19
Software Engineering Economics (Boehm), 100
Software Engineering Institute Web site, 299
“Software is Different,” 331
Software Project Survival Guide (McConnell), 59, 173, 475
Software Requirements (Wiegers), 49, 53, 173, 518
Software Research’s Quality Techniques Newsletter archive, 331
Software System Testing and Quality Assurance (Beizer), 95, 173, 268, 441
Software Techniques (Beizer), 406
Software Test Automation (Fewster and Graham), 101, 112, 173, 179, 508, 313, 415, 499
Software Testing and Quality Engineering, xix, 66, 398
Software Testing and Quality Engineering Web site, 199
Software Testing in the Real World (Kit), 28
Software Testing Techniques (Beizer), 25, 48, 53, 173, 196, 296, 300, 314, 399
Source code control engineer, 349, 526, 527
Source code control system, 327, 526, 527
Specialization in test team, 264–266
Specialized training, 242
Speedy Writer test team, 14–18
Splaine, Steve, 372, 389
Sponsors, 4, 521, 527
Stable quality risk analysis, 84
Staffing, 86–87
Stakeholders, 4, 527
convincing to participate in risk, 62–63
representing interests of classes of customers or users, 61–62
selling on quality risk management, 63
test plan, 197
Standardization, 490–497
State graph, 281
Static tests, 513
Status testing, 173
“Stop Destroying My Team with Bad MBOs: A Manager’s New Year’s Resolution,” 506
strings utility, 343
Substitution, 108
Subsystem testing, 10, 521, 528
Sumatra project, 14–18
  alpha test, 17–18
  assignment of levels of testing, 49
  attack on Big Build, 360–368
  beta test, 17–18
  big bug in Big Build, 392–398
  Big Build test results, 422–440
  budgets, 113–119
  change management, 464–474
  code-and-fix, 17
  code-and-fix tar pit, 16
  context of effort, 18
  convincing managers about budget, 126–129
  Dashboard, 422–440
  decomposing test project into phases, 70–75
  dependencies, 76
  developing work-breakdown-structure and estimated schedule, 80
  document management subsystem, 36–37
  estimating risk, 70–80
  first incremental delivery target, 16
  Incremental development model, 14, 16
  integration testing, 17, 72–73
  pertinence, 18
  quality risk analysis, 36–44
  quality risks, 73
  resolving discrepancies in test subproject schedule and project schedule, 78–79
  resource assignments, 76
  retrospective test team report, 515–517
  reviewing work-breakdown-structure and schedule, 79–80
  skills growth, 249–252
  test coverage analysis, 301–305
  test plan, 155–170
  test release, 327–334
  test strategy, 171, 173
  test system, 275–285
  test teams, 223–232
  Waterfall releases, 16–17

*Surviving the Top Ten Challenges of Software Testing* (Perry and Rice), 501

System being maintained, xxxiv, 528
System development lifecycle, 6, 18–23, 529
System development process and bug reports, 412
System Engineering group, 14
System environments and installable releases or packages, 335
System lifecycle, 6, 529
  positive ROI (return on investment), 123
testing process within, 8
System of systems, xxxiv
System quality readiness, 194–195
System test
  continuation criteria, 164
  entry criteria, 163
  exit criteria, 164, 196
  incomplete and unrealistic releases, 351–352
  Incremental approach, 74
  staffing needs and pass durations, 74–76
System testing, 10, 529
System under development, xxxiv, 528, 529
System under test, xxxiv, 412, 528
Systems, xxxiv, 528
  endangering quality, 5
  expectations of quality, 3
  mission-critical, 291–291
  safety-critical, 291–291
  unfit for testing, 195

**T**

Taguchi methods, 295
Tasks, 70, 235–236, 529
Taylor series calculation, 286
Teams
  high-quality process for all participants, 255–256
  performance and estimates, 132
  performance estimates, 132
  risks, 81
Technical expertise, 232–233
Technical risk, 62, 299
Technical skills, 233, 529
Tejas Consulting Web site, 300
Test and Measurement.com, xix
Test artifacts, 182, 529
“Test Case Summary Plan (1Shift)” worksheet, 181
“Test Case Summary Plan” (2Shift) worksheet, 181
Test cases, 277, 529
automated testing harness, 336
data sets shared across, 309–310
dependencies between, 309–310
as documentation, 99
documenting, 402
failure, 315
linking results to bug reports, 400
multiple bugs in single step, 315
number of conditions, 314–316
obscuring other failures, 315
people creating, 374
precision, 288
preparations for running, 308–309
quality risk coverage, 77
speed of conclusion, 308
time and resources available to design and implement, 292
tracking, 383
Test conditions, 277, 298, 530
Test coverage, 530
analysis, 127
analysis techniques, 297–300
behavioral (or black-box) coverage, 298
behavioral coverage assessments, 299
behavioral test coverage, 300
black-box coverage, 304
code coverage, 300
cyclomatic complexity, 299–300
gray-box coverage, 304
present or absent, 299
quality risk coverage, 298
structural test coverage, 300
Sumatra Project, 301–305
technical risk, 299
Test cycles, 73, 122, 530
Test dashboard, 441–442
Test data, 310, 530
Test design
appropriate documentation, 287–293
assessing quality of existing test system, 320–321
balancing coverage, schedule, and budget, 319–320
built-in profiling tools, 303
capable tools for assessment of quality, 305–305
code complexity tool, 303
code coverage tool, 303
combinatorial explosions, 293–296
improving, 320–322
managing test system, 307–311
number of conditions in test case, 314–316
preventing bugs, 314
retesting third-party components, 316–317
selects right techniques, 311–314
test oracles, 285–287
vaguely defined requirements, usage profiles, and environments, 317–319
“Test Design: Developing Test Cases from Use Cases,” 296
Test development
automated tests, 89
building custom test tools, 89
dependencies between activities, 87
estimating projects, 87–91
order of, 87–88
rules of thumb, 90–91
special data, 89
tentative approval for estimate, 87
test execution, 94
testing tests, 90
updating existing test scripts, 89
verification tasks, 90
“Test Effort Estimation,” 81
Test engineers, 235, 306, 530
Test environment configurations, 183–185
diagram, 156–157
estimating projects, 91–92
Test environments, 274, 530
administrator permissions, 378
changes to, 184
complexity, 92, 350
customer-like configurations, 183
debugging, 184
details, 155–156
executing resources, 181–183
management policies, 184
managing testware and other documents, 185
partitioning, 340
password and access control, 184
people outside test team, 184
prioritizing use of resources, 182
problems with, 183–184
service-level agreements, 184
tracking test items used, 185
trivial, 92
Test escape, 179, 530
Test estimation, 81
Test execution
accommodating holidays, cultures, and vacations, 381–382
arrival of certain components, 95
assessing, 382
bite-sized pieces, 382
blind spots, 369
capturing historical information, 382
challenges, 379–382
communication between test team members, 379–380
complexity of process, 291
correctly interpreting test results, 372–375
criteria for phases, 194–196
defect removal models, 94
educated guess about nastiest bugs, 368
efficiency, 377–378
estimating projects, 92–95
exhibiting proper attitudes, 375–377
exploratory testing, 93
extended shifts, 379–381
finding problems, 374
finding scary stuff first, 368–370
Golden Candidate test release, 94
outsourced testing, 379–381
pass duration, 94
process, 355–356
producing, gathering, and disseminating valuable information, 370–371
quality risk analysis, 368–369
test case count, 93–94
test development, 94
test tracking worksheets, 357–360
tight time windows, 291
tracking test cases, 383
Waterfall system development lifecycle, 95
Test execution process, 274, 530
Test granularity, 174, 530
Test group project-based assignment strategy, 113
Test level, 27, 531
Test management tools, 310, 531
Test managers
advocate and communicator of team, 26
conflict of interests, 26–27
dependencies, 26
documentation, 449
explaining realities of testing, 123–124
formalizing test process, 494–495
people management skills, 26
perceived effectiveness, 32
understanding financial implications of actions, 144
Test oracled, 147, 531
Test oracles, 285–287
Test organization, 24
Test passes, 73, 531
Test phase, 27, 73, 531
Test plan, 85
brevity, 203–204
bug report handoffs, 162
building consensus, 214
canned, 207
Test plan, continued
challenges, 207–214
clear criteria for phases, 194–196
collaborative processes with team, 160, 162
commitment, 197–199
common expectations, 197–199
compromising with key player, 210
consensus, 197–199
context, 160, 214
contingency and mitigation plans, 166, 167–169
details of test environment, 155–156
documentation, 202–204
entry and exit criteria, 162
escalation for blocking bugs, 162
executive summary, 170
failure to identify support personnel, 212
flexibility and creativity, 200–202
government regulation, 213–214
hardware usage plan, 156–157
improvements, 214–215
initial estimates, 156
integration test, 163
key considerations, 170–187
key players not supporting, 209–213
learning curve, 199
legality, 213–214
major milestones, 159
mentors, 159
meshing planning processes, 205–206
mitigating risks, 166
multiple, 203–204
objective of process, 214
opportunities to catch errors, 204–205
outsourcing plan, 208–209
potential points of contention, 197
preventing surprises, 197
regression risk management strategies, 176–180
repeating from previous testing, 159
scope creep, 204–205
selecting strategies for testing, 171–176
smooth team handoffs, 160, 162
stakeholders, 197
status reporting, 162
subproject readiness, 194–195
subproject risks, 185–187
Sumatra Project, 155–170
system quality readiness, 194–195
system test, 163–164
tasks and reminders, 166, 170
“TBD” markers, 198
template, 207–208
test environment configuration diagram, 156–157
test environment configurations, 183–185
test environment executing resources, 181–183
test release process, 162
test strategies, 155
test systems, 183–185
testers, 156
training, 159
what if thinking, 201
work-breakdown-structure, 159
written, 214
Test procedure, 277, 312
Test Process Improvement (Koomen and Pol), 81, 490
Test processes, 498
addicted to testing, 507–508
advantages and disadvantages of changing, 513
challenges, 501–508
compliance with, 495
continuous learning and inspiration, 500–501
costs, 495
effectiveness, 487
efficiency, 487
employing appropriate strategies, 497–499
fine-tuning big processes, 509–511
flexibility and intelligent tailoring, 496
formalizing and standardizing, 291
identifying and removing big process obstacles, 511–513
identifying gaps, 513
immature development or maintenance processes, 503–504
implementing change, 513–514
information about other lifecycle processes, 6
institutionalizing change, 514
length of, 6
measuring what we can't define, 502–503
perfecting, 6
performing, 6
pervading projects, 487–489
planning, 6
planning change, 513
preparing, 6
providing valuable, economical services, 485–487
reducing efficiency, 495–496
return on investment, 7
successfully introducing tools, 499
test team relationships, 506–507
unrealistic expectations, 504–506
Test professionals, 220, 531, 532
obstacles, 122
perceived effectiveness, 32
Test project
decomposing into phases, 70–75
plan, 125
“Test Release Processes,” 326, 461
Test releases, 325, 327, 353, 520, 531
80/20 rule, 337
balancing benefits against risk, 352
build suitable for testing, 335
building from carefully selected, coordinated, and documented content, 344–347
challenges, 547–352
complexity of system, 348–350
coordinating, 339
defined, customer-like install and uninstall processes, 340–342
fatally flawed, 341
incomplete and unrealistic system and acceptance, 351–352
installable, improved, and stable, 335–337
installing, 337–338
logging bugs, 342
management, xxvi, 325
naming, 342–344
order of installation, 337
outdated, 338
ownership of processes, 347–348
predictable, timely, and regular, 337–340
process, 326
release ID, 343–344
release identifier, 342
revision control, 336
revision number, 342
selecting content, 330–331
software configuration management, 336
Sumatra project, 327–334
tentative changes, 346
Test results reporting
accuracy, 442
appropriate mechanisms and documents, 448–450
appropriate size steps, 458
articulating results, 445–446
asking stakeholders thoughts, 458
average number of bug reports, 425
bad news to report, 453–457
challenges, 451–458
dispassionate presentation, 458
effective communication, 442, 445–446
e-mail, 448
as ever-changing complex story, 457–458
effective communication, 442, 445–446
e-mail, 448
as ever-changing complex story, 457–458
executive summary, 425
graphically representing information, 442–443
graphs and charts, 450
handling presentations, 452–453
influencing future progress, 444–447
Internet postings, 450
keeping overhead low, 458
Test results reporting, continued
monitoring effectiveness, 458
passion for quality, 458
process, 420–421, 440–451
relating test status to project progress, 444–447
right channels for transmitting information, 444
showing examples, 458
showing trends and destinations in quality over time, 447–448
slide-show-style presentation, 448
snapshots, 448
tailoring to each listener, 450–451
test dashboard, 441–442
useful, credible, and timely information, 441–444
Test scripts, 277, 312, 416, 531
Test set, 356, 531
Test specialist, 235, 531, 532
Test strategy, 531
agile methodologies, 172
based on top-down test designs, 171
bottom-up test designs, 171
exploratory testing, 172
help with, 175
quality risks, 155
structural analysis, 172
Sumatra Project, 171, 173
test automation, 172–173
type of test techniques, 173–174
Test subprojects, 66, 76–78, 531
budget, 186
dependent tasks, 76–78
features, 185
quality, 186
readiness, 194–195
resource assignments, 76–78
risks, 185–187
schedule, 185, 186
Test suite, 277, 532
Test systems, 112, 155, 183–185, 532
appropriate documentation, 321
assessing and harmonizing tester skills, 321–322
assessing quality, 320–321
avoiding waste, 307–308
buying right tools and environment, 308
complexity and test releases, 348–350
control-flow-based testing technique, 280
critical risks, 273
degree of product, 291
design, 275–285
document management system, 279–280
dumb monkeys, 280, 281
effective, 307
efficiency of use, 308
hierarchical storage management systems, 281
implementation process, 275–285
insufficient, 308
load testing, 280
long-range plan, 321
managing, 307–311
mimicking, 286
precisely documented, 288
predicting correct behaviors, 286
purchasing useless elements, 308
quality risks, 275, 277
reliability, 308
reuse efficiency, 308
running test case, 308–309
soap opera approach, 279, 280
state graph, 281
stubs, 283–284
Sumatra Project, 275–285
test environment, 274
test execution process, 274
testware, 274
usability, 311
virtual-user licenses for loading testing tool, 281
Test teams
accommodating holidays, cultures, and vacations, 381–382
accurately defining job, 256–257
advance notice, 381–382
advertising position, 224
allowing appropriate specialization, 264–266
application domain expertise, 238
appropriate and fair salaries, 262–263
audition interview, 228, 231
backwater or dumping ground, 240
building, 220–232
candidates, 220
career path, 227, 248
challenges, 266–271
communication between members, 379–380
compatibility, 272
consultants, 267, 269–271
continual turnover, 240
contractors, 269–271
credibility, 267
customers’ power users, 234
differentiation of skills and abilities, 236
dynamic skills, 248
experienced testers, 264
filtering candidates, 227
financial benefits from early involvement, 106
good building process, 254–266
growing skills, 219
growth, 248
hiring, 220
identifying and managing critical skills, 232–235
incessant and involuntary turnover, 261
independent, 24
in-person interviews, 227–228, 231
integrated into project team, 24
internal failure, 116
job descriptions, 224, 259–260, 263
lesser-paid, lesser-skilled members, 235
levels, 257
logistics, 380
long-term and short-term benefits, 254
long-term career path, 257–262
low-level employees, 222
managing inward, 31
military organizations, 222–223
morale, 26
multiple projects, 265
must-fix bugs, 116–117
mutual satisfaction of all parties’ objectives, 254–255
new hires, 239–240
offer letter, 231
opportunities for specialization, skills growth, and career paths, 26
organization, 265
orientation, 231
participation in change management, 474–475
partitioning, 340
permission to hire, 223
phone interviews, 227, 248
postinterview discussions, 231
project-based, 265
relationships within, 506–507
release notes, 346
reorganizing, 265–266
repetitive tasks, 235–236
resume, 229–231
resumes, 224
retaining, 263–264
reviewing and ranking resumes, 248
reviewing skills, 223–224
right test professionals, 219
as risk officer, 59
rotation, 238–239, 261–262
salary range, 227
scheduling, 380
service-based, 27–28, 265
services to multiple projects, 26
skills analysis, 225–227
skills assessment and management, 272
skills growth, 249–252
skills management, 252
skills profile, 234–235
skills-based, 265
staff members you can’t promote, 240
staffing variations, 238–240
Sumatra Project, 223–232
support, 380
temporary assignment, 238
test engineer, 235
test manager, 26
test technicians, 235–238, 257
Test teams, continued
  titles and roles, 258
  training, 263–264
  unresolved issues, 256
  value added by, 13, 18
  variations on building, 222–223
  well-rounded group, 234
  white elephants, 268–269
  win-win philosophy, 254–256
Test technicians, 74, 86, 235–238, 257, 532
Test tracking worksheets, 357–360
Tester track, 261–262
Testers, 220, 532
  ability to focus, 244
  accepting generalist and supporting role, 244–245
  advocating quality, 244–245
  attitude, 244–245
  balanced curiosity, 244, 376
  bringing bad news to development team, 245
  bug reports, 346
  career path, 254, 272
  certification training, 242–243
  comparing attributes, 13
  competence, 267
  continuing education, 242
  disagreements or different understandings, 376
  education, 241–242
  exhibiting proper attitudes, 375–377
  experienced, 264
  explaining expected behavior, 410
  extended shifts, 379–381
  focus, 376
  formalizing test process, 494–495
  group synergy, 25
  hardworking, 245
  honesty in self-assessment, 253
  interview questions, 245–248
  as junior programmers, 28, 30
  levels of skills required, 266–267
  motivational concerns, 292
  need-to-know information, 266–267
  organizing, 24–31
  outsourced testing, 379–381
  perceived as competent, 267
  perverse incentives, 26
  professional pessimism, 244
  versus project contributors, 12
  reasonable user expectations, 286
  self-righteous about quality, 122
  skills, 243–244, 266, 292
  specialized training, 242
  spending inappropriate amount of time on problems, 374–375
  test plan, 156
  testing tools vendors certification, 243
  training, 242
  understanding of customers’ and users’ experiences of quality, 273
  undervaluing degree of skill required, 261
  unresolved issues, 256
Tester-to-programmer ratio-based estimates, 82
Testing, 8, 532
  adapting to change, 54–56
  alpha, 17
  balanced, 41
  behavioral, 173
  beta, 17–18
  black-box, 173
  boundary with debugging, 406–408
  business case for, 112–113
  business realities, 122
  as career-killing dead end, 257–258
  collaborative process, xxix–xxx
  communicating information, 31–32
  comparing, 390
  component, 17
  compromise on scope, 148–149
  context, 214
  Department of Defense 2617A standard, 398
  different phases, 8
  discussing corporate activities, 18
  elements of, 13
  extensive, 41
finding potential bugs before, 52–53
fitting into bigger picture, 122
generating information, 31
glass-box, 173
higher-level phases, 24
history, 11–12
history and bug reports, 402
importance of, 122
ineffective and inefficient, 338
insight into quality, 110–111
internal processes, xxix–xxx
investing in, 3–5
as job with undesirable attributes, 245
kind of investment, 138–144
live, 174
mitigating risk, 41
none or minimal, 11–12
operational context, 11–14
opportunity, 41
organizational context, 11–14
organizing, 24–31
pervasive, 487–488
point of diminishing returns, 104
positive ROI (return on investment), 123
providing value, 28, 30
reductions in scope of, 97
reporting observed bugs, 41
requirements for standardized, 11
ROI (Return on Investment), 98–113
selling, 124
skills, 209, 532
solid, organized, 389
stakeholders and key quality, 36
status, 173
structural, 173
studying existing, 16
within system development lifecycle, 18–23
tools, 532
understanding role and value, 33
unexpected behaviors, 377
unimportant areas, 35
value of effective and efficient, 11
well-formed work-break-down-structure, 67
white-box, 173
_Testing Applications on the Web_ (Nguyen), 174
_Testing Computer Software_ (Kaner, Faulk, and Nguyen), 48, 173, 174, 268
_Testing Computer Software, Second Edition_ (Kaner, Faulk, and Nguyen), 369, 391
Testing engineer, 86
“Testing in the Dark,” 319
_Testing Object-Oriented Software_ (Binder), 25
Testing plan benefits, 118–119
_Testing Practitioner, The_ (van Veenendaal), xix, 388
Testing process, 8–11
accepted, 489
adding value, 35
checklists, 8, 9
collaborative, 488
communicating, 488
concurrent, 487
cross-functional, 488
cross-pollinating testing and development activities, 488
dependencies, 23
easily understood, 489
effectiveness, 489
efficiency, 489
formality, 490–497
within lifecycle, 8
mature, 489–490
multiple instances, 8–9
optimizing, 489–490
organized, 489
peer-level stakeholders, 13, 18
sequencing, 23
standardization, 490–497
Testing skills, 232–233, 233
Testing tools, 310
Testing tools vendors certification, 243
Testing Web Security (Splaine), 417
Testing.com Web site, 28, 369
Tests
  behavioral, 313
  beta test sites, 282
  cascading failures, 308
  coarser-grained, 174
  correctly interpreting results, 372–375
design, 275
design-based, 174
dynamic, 313
focusing effort, 53
fulfilled, 359
implementation, 275
lowest possible overhead, 378
manual and automated, 312
mapping back to some property of system under test, 298
planning, 383
relating status to project progress, 444–447
reliability, 308
repeating, 179
reusability, 313
state of, 357–358
static, 313
structural, 313
studying and applying advances, 321
that mitigate risks and ROI (Return on Investment), 107–110
unfulfilled, 359
Tests results reporting, 419
Testware, 274, 288, 532
Third-party components, retesting, 316–317
Three-Point, 90
Time-to-market, 119
T-MAP (Pol), 49
tmp file extension, 403
Top-down estimation, 81
Tracing internal information, 403
Training
test plan, 159
test team, 263–264
test technicians, 237
testers, 242
Trial and Death of Socrates, The (Plato), 136
Trudeau, Garry, 119
True Odds (Walsh), 57
Twain, Mark, 204

U
Unambiguous testware, 288
Unit testing, 10, 25, 28, 533
Unrealistic expectations, 504–506
Upward management, 32–33
Usability Engineering (Nielsen), 174
Usability testing budgets, 113
User acceptance testing, 10, 519, 533
User surrogates for formal or informal test, 24
Users, 4, 533
formal or informal test, 24
quality risk analysis, 45
“Using Monkey Test Tools,” 280
“Using Oracles in Test Automation,” 287

V
V Model, 18, 19, 21, 174, 533
design and implementation of test system, 23
incremental test releases, 340
Vacations, 381
Value delivered, 3
Value of risk mitigation, 117–118
Vasquez de Coronado, Francisco, xxii
Verbal and written communication, 232
Version, 342, 520, 533
Version number, 342, 520, 533
Virtual-user licenses for loading testing tool, 281
Visual Display of Quantitative Information, The (Tufte), 443, 446
Visual Explanations (Tufte), 443, 446, 452

W
Wambke, Alyn, 419
Wasting time and money on status reports, 443
Waterfall model, 16–17, 18
incremental test releases, 340
Waterfall System Development Lifecycle, 19, 95, 533

*Web Testing Handbook, The* (Splaine and Jaskiel), 292

Weekend or holiday shift, 379

Weller, Ed, 102–103

What if thinking, 201

“What is Quality,” 502

“When Helping Doesn’t Help,” 508

White elephants, 268–269

White-box tests, 25, 173, 527, 533

Wideband, 90

*Winning at Project Management* (Gilbreath), 257

Win-win philosophy, 254–256

Workaround information, 404

Work-breakdown-structure, 127

automated software estimation tools, 81

cHECKING schedule, 79

decomposing test project into phases, 70–75

developing, 67, 80

index-card approach, 71

logistics behind tasks, 85

not meeting criteria, 146

reviewing, 79–80

rough estimations, 81–82

tEST development section, 159

test plan, 159

updating, 283

Working files, 403

Written test cases, 291