

# Index

## A

Acceptance tests. *See also* Executable acceptance tests  
asking questions when developing, 12–13  
automating, 64, 67, 84, 90–91, 133–137, 194, 208  
coupling between, 128–130  
courage and, 27, 28  
criteria for, 86, 181  
defining, 26, 44, 55–60  
details, 94–95, 99–108  
educating the customer about, 180  
estimates, 61–69, 71–75  
feedback and, 5, 26, 28  
files, 120–121  
frequency of, 177–180  
hidden assumptions and, 58  
high-level, 61–69, 255–259  
iteration planning and, 80  
missing, 195–196

organizing, 119–124  
preparation time for, 64, 65, 66  
quality assurance and, 32, 93–97  
quick-and-dirty approach to, 63–64  
for retail Web sites, 35–36  
retrospectives and, 190, 191  
road-trip metaphor and, 40, 61  
running, 147–156, 177–174  
self-verification and, 144  
separating, from development, 87–91  
in spreadsheets, 121–122  
system state and, 126–128  
tasks, identifying/estimating, 85–87, 90–91  
templates, 56–59  
testability and, 204–205  
test infrastructure and, 88, 90–91, 221  
time required for, 44, 61, 63–69, 87

Acceptance tests, *continued*

- unit tests versus, 19–21
- in user stories, 43–44
- version control for, 119–120
- writing, 100, 108–117, 230

## Accessories, 216–218

## Accountants, 18, 38

## addTask() method, 273

addTask module, 266–267,  
282–283

## Agile-testing mailing list, 246–247

## Air Force (United States), 27

## Algorithms, 56, 106

## AllTests class, 154–155

## AllTests.java, 154–155

## Application

- program interfaces (APIs), 84
- test classes, 148–149, 160–166

## Arguments, passing, 165

## assertEquals() method, 112

assertFalse() method, 110–111,  
148–150, 163assertTrue() method, 110–113, 144,  
148[150]

## Assumptions

- examples of, 49–54
- identifying, 43–45, 47–54,  
251–258, 262
- iteration planning and, 80
- XTrack application and, 53–54

## Automation

- acceptance tests and, 64, 67, 84,  
90–91, 133–137, 194, 208
- accessories and, 216–218
- baseline comparison method and,  
142–144, 145
- basic description of, 133–137,  
139–145
- capture/replay, 140, 170

data-independent tests and,  
142–143

## design spike, 86

## direct-call testing and, 166–167

## evolving tools and, 208

## legacy systems and, 203

## mastering, 133–134

## metrics and, 219–220

## modular tests and, 141–142

## self-verifying tests and, 143–144

## test infrastructure and, 86

## time estimates for, 44

## tools for, 211–212

## unit testing and, 7, 9, 18, 245

**B**

## Back button, 101

Bad path, 101–102, 104–108,  
268–270Baseline comparison method,  
142–144, 145, 170

## Beck, Kent

- on the comparison between XP and  
driving a car, 20

## core values of XP and, 4

## on feedback, 26

## on internal and external quality, 95

## introduction of XP by, 3

on the Programmer’s Bill of Rights,  
31

## on simplicity, 25

## on teamwork, 23

## on testers, 43

## Bells, ringing, 217

## Best practices, 30, 222

## “Big-bang” integration, 120

## Bill of Rights

## for programmers, 31

## for testers, 31–32, 78–79

- Blended practices, 229–232
- BoldTech Systems, 159, 166–167
- Bootstrap time, 85
- Brainstorming, 182–183, 194, 244
- Bug(s). *See also* Defects
  - testing resources wasted on, 7
  - tracking systems, 182, 209
- Build scripts, 84
- Business
  - experts, 241–245
  - logic, 194
  - problems, solving, 49–54, 252, 257–260
- “Button push” action, 166
- C**
- Canoo Web Test, 208
- Capture/replay-style automation, 140, 170. *See also* Automation
- CASE tools, 209
- Celebrating success, 185, 217
- CGI (Common Gateway Interface) scripts, 159. *See also* Scripts
- “Challenges for Analysts on a Large XP Project” (Schalliol), 235, 236, 237–238
- Checks and balances, 18–19
- Clark, Mike, 246–247
- Classes
  - application test-interface, 148–149, 160–166
  - refactoring, 162–163
  - tool-specific, 164–166
- Coaches
  - provision of snacks by, 222
  - support of testers by, 228
  - tester and, similarities between, 20
- Cockburn, Alistair, 23
- Code. *See also* Scripts (listed by name)
  - collective ownership of, 6, 135, 222
  - compiling, 90, 148, 149, 151
  - standards, basic description of, 5
- Collins, Steve, 42
- Communication. *See also* Feedback; Meetings
  - basic description of, 4–5, 24–25
  - with customers, 4–5, 24–25, 100–101
  - grade cards and, 186
  - iteration planning and, 80–81
  - as one of the four core values of XP, 4–5
  - with programmers, 100–101
  - test infrastructure and, 88
  - tools, educating the customer about, 181
- Compilers, 90, 148, 149, 151
- Concurrent users
  - acceptance tests and, 62, 67
  - hidden assumptions and, 52
  - number of, 52, 96
  - quality assurance and, 52, 96
- Constructors, 148, 152, 163
- Continuous integration, 5, 7
- Contractors, 12–13, 18. *See also* Remodeling project
- Core values. *See also* Communication
  - basic description of, 4, 23–28
  - courage, 4, 6, 25, 27–28, 30–31
  - feedback, 4–5, 26–29, 219–220, 242, 244
  - simplicity, 4–6, 25, 186
- Coupling, between tests, 128–129
- Courage
  - basic description of, 6, 27–28
  - embracing simplicity and, 25
  - fears held by testers and, 30–31

- Courage, *continued*  
as one of the four core values of XP,  
4  
tester role and, 30
- Crashes, 37, 96
- create() method, 129, 153
- createUserID() method, 127–128,  
153, 160, 163, 165
- createUserID module, 170–171,  
173–175
- Creativity, in writing tests, 105
- Customer(s)  
acceptance tests run by, 178–179,  
246  
as business experts, 241–245  
communication with, 4–5, 24–25,  
100–101  
defect reports and, 209  
educating, 180–181, 183  
expectations of, grade cards and,  
186  
hidden assumptions and, 47–54  
interviews with, 100–101  
iteration planning and, 78  
non-critical tests and, 104–105  
“picking the brains” of, 100  
quality assurance and, 94–95  
role of, in XP, 29–30, 36  
test environments and, 221  
turning software over to, 197–198  
weekly meetings with, 237  
working with, overview of,  
237–238
- Cyclomatic Complexity Number, 220
- D**
- Databases, 56–59, 89, 101, 159  
system state and, 126  
testability and, 204–205
- Data-independent tests, 142–143
- Deadlocks, 36
- Debugging tests, time estimates for,  
44
- Defects. *See also* Bugs  
acceptance criteria and, 181  
code missed by direct calls and, 158  
educating the customer about,  
180  
issues that look like, but are outside  
the scope of iterations, 181  
logging, 196  
maintenance and, 196  
management of, 181–182  
reporting, 86, 179–180, 209  
tracking systems, 182, 209
- delete() method, 129, 153
- deleteTask module, 266–267
- deleteUserID() method, 128, 153,  
160, 163, 165
- deleteUserID module, 171–175
- Development  
infrastructure, iteration planning  
and, 79  
separating acceptance testing from,  
87–91  
test phases after, 227–228  
waterfall, 106, 230, 241
- Direct call(s)  
code missed by, 157, 158  
interface, refactoring, 161–162  
test automation, 166–167
- Discipline, importance of, 225
- Documentation  
grade cards and, 186  
quality assurance and, 38  
requirements, 238–239, 242  
retrospectives and, 186, 190  
test plans, 238–239, 243–244

**E**

Egos, 135

Error(s). *See also* Bugs; Defects  
 acceptance tests and, 103  
 conditions, 103  
 handling, 52, 101  
 hidden assumptions and, 52  
 requirements for, 101  
 screens, quality criteria and, 37, 96

Estimates  
 for acceptance tests, 61–69, 71–75  
 for automation, 44  
 grade cards and, 186  
 importance of, 71–72  
 improving the accuracy of, 71–75  
 for tasks, 80–81, 83–92  
 velocity, 44, 85, 274–275

Executable acceptance tests,  
 110–113, 120–121, 144,  
 147–156. *See also* Acceptance  
 tests  
 combining, into test suites,  
 154–155  
 expanding coverage of, 158–159  
 frequency of, 177–180  
 linking, to application test classes,  
 148  
 missing, 195–196  
 running, 147–146, 150–152,  
 177–174  
 writing, 110–113

extends clause, 152

*Extreme Programming Explained*  
 (Beck), 20, 23, 43, 222  
 facilities strategy in, 215–216  
 feedback in, 26  
 internal and external quality in, 95  
 new perspective fostered by, 198  
 simplicity in, 25

test-first programming in, 244–245  
 unit tests in, 244–245

*Extreme Programming Installed*  
 (Anderson, Hendrickson, and  
 Jeffries), 179–180, 217

**F**

Facilities, strategy for, 216–216

Far Side cartoons, 135–136

Fears, held by testers, 30–31. *See also*  
 Courage

Feedback. *See also* Communication  
 acceptance tests and, 28  
 as one of the four core values of XP,  
 4–5  
 basic description of, 26–27  
 continuous, 29  
 gathering requirements and, 242  
 metrics and, 219–220  
 “smoke” testing and, 244

FillOutForm module, 141

Forward button, 101

Fowler, Martin, 31, 185

Framework, MDS (Modular Data-  
 driven Self-verifying) test  
 framework, 170–175

Functional tests  
 onsite customers and, 4  
 tasks for, identifying/estimating,  
 85–87

**G**

Gaye, Ben, III, 249

getField utility, 173

GET method, 170

getTestCase() method, 171

getTestCaseName() method, 171

GNU license, 220

Grade cards, 186, 218

Gregory, Janet, 220  
 gTestCaseFile parameter, 171  
 GUI (graphical user interface), 101,  
 151–152, 159, 168, 204, 281.  
*See also* User interface  
 GUITAR tool, 168, 281–282

## H

Hacker attacks, planning for, 102  
 Handoffs, 186  
 Happy path, 101, 103, 107–108,  
 268, 269  
 HTML (HyperText Markup  
 Language), 166  
 HTTP (HyperText Transfer  
 Protocol), 65, 159–160, 171,  
 173, 208, 212  
 HTTPUnit, 208, 212

## I

IDE (integrated development  
 environment), 79, 188  
 import jWebART.\*; statement,  
 165  
 import xtrack.\*; statement, 162, 163  
 Individuals, characteristics of,  
 importance of, 23–24  
 init() method, 171  
 Integration tests  
 lack of adequate, 8  
 road-trip metaphor and, 40  
 using short, 193  
 invoke() method, 165  
 Iteration(s). *See also* Iteration  
 planning  
 display/update information about,  
 107  
 feedback and, 26–27  
 grade cards and, 186

retrospectives at the end of,  
 185–186  
 testing at the end of, 179  
 Iteration planning. *See also* Planning  
 acceptance tests and, 62, 93, 230  
 basic description of, 77–81  
 development infrastructure and, 79  
 educating the customer about, 180  
 enhancing communication and,  
 80–81  
 identifying tasks during, 79–80  
 for large or multi-location projects,  
 239–240  
 release planning and, 78  
 road-trip metaphor and, 40  
 role of testers in, 78–81  
 test infrastructure and, 80, 83–92  
 IterationTestStory.xml, 275–276

## J

Java. *See also* .java scripts  
 acceptance tests and, 110–113,  
 120–121, 148–152  
 advantages of, 110  
 compilers, 148, 149, 151  
 interpreter, 151  
 Native Interface (JNI), 157  
 packages, 164, 170, 171  
 release planning and, 42  
 server pages (JSP), 166, 203  
 testability and, 293  
 use of, for examples in this book,  
 110  
 JavaNCSS, 220  
 JavaScript  
 testability and, 293  
 WebArt and, 160  
 .java scripts  
 AllTests.java, 154–155

- LoginStoryTest.java, 120, 122, 151
  - TaskStoryTest.java, 279
  - TestLoginStory.java, 121
  - UserIDStoryTest.java, 152–153
  - XTrackDirectInterface.java, 161–162
  - XTrackTest.java, 149, 153, 162–163, 279–280
  - JBuilder, 188
  - Jeffries, Ron
    - on roles, 29
    - on writing tests at the GUI level, 204
  - JNI (Java Native Interface), 157
  - JProbe, 220
  - JSP (Java server pages), 166, 203
  - JUnit
    - constructors and, 148
    - direct-call test automation and, 166–167
    - as an evolving tool, 208
    - executable tests and, 151
    - metrics and, 220
    - regression tests and, 195
    - running tests and, 147–156
    - tests under, combining test tools with, 209
    - Web site, 110, 111
  - Just-in-time planning, 41
  - jWebART package, 164, 170, 171
  - jWebARTInterface, 171, 173
- K**
- Kini, Natraj, 42
  - Knowledge sharing, 190
- L**
- Languages, programming
    - compatibility of, with tests, 111–112, 157
    - selecting, 110
  - Leadership. *See also* Coaches
    - correcting problems and, 38
    - support of testers by, 228
    - test infrastructure and, 88
  - Legacy systems, 203
  - Lights-out test design, 105–106
  - Login
    - acceptance tests and, 111–114
    - concurrent, number of, 37, 52, 62, 67, 96
    - executable tests and, 148–150
    - modular tests and, 142
    - system state and, 128
    - validation criteria for, 173–175
  - login() method, 111, 142, 148–150, 158, 165
  - login module, 171–175
  - LoginStoryTest class, 120–121, 148–152, 154–156, 168, 279, 281, 284
  - LoginStoryTest.java, 120, 122, 151
  - “Look and feel,” testing for
    - consistent, 136
- M**
- MCabe metric, 220
  - Maintenance, 196
  - Managers. *See* Coaches; Leadership; Teams
  - Manual tests
    - disadvantages of, 133–138
    - divisiveness of, 125
    - self-verifying tests and, 144
    - unreliability of, 134
  - Marick, Brian, 210

MDS (Modular Data-driven Self-verifying) framework, 170–175

Meetings. *See also*

Communication

retrospective, 185–192, 209, 245–246, 284–285

standup, 186, 188, 190, 191, 218–219

weekly, with customers, 237

Merriam-Webster Web site, 24

Metrics, 219–220

Microsoft NetMeeting, 239, 240, 286

Milestones, 238

“Missing door” syndrome, 12–14, 43–44, 51–52

Mistakes, learning from, 185

Modularization, 140–142

Morale, 23

multiSearch module, 90, 91

Murphy’s Law, 14

myOutCome variable, 172

## N

Narrowing things down, concept of, 77, 82, 265

NCSS (Non-Commenting Source Statements), 220

NetMeeting (Microsoft), 239, 240, 286

Newsgroups, 210

Noncritical tests, 104–105

## O

Office space, 215–216

Open-source tools, 208, 209

Optional tests, 104–105

outCome variable, 173

## P

Pair programming

basic description of, 5

estimating time for, 80

grade cards and, 186

insuring that XP practices are really using, 9

iteration planning and, 80

running tests and, 147

switching pairs and, 188, 189, 191

unit tests and, 19–20

Parameters, 142, 165, 171, 172

Passwords

acceptance tests and, 111, 113–115, 150, 153

login module and, 172

modular tests and, 142

Performance testing, 79

Pettichord, Bret, 210, 241

Pettichord.com Web site, 210

pIn parameter, 172

Planning. *See also* Iteration planning;

Planning Game; Release planning

advantages of, 236

educating the customer about, 180

just-in-time, 41

*Planning Extreme Programming* (Beck and Fowler), 20, 31

Planning Game. *See also* Planning

acceptance tests and, 99

basic description of, 6

courage and, 6

defining quality and, 35–37

including time estimates for testing in, 44

role of testers in, 13–14, 42–43

POST method, 170

pOut parameter, 172

Programmers. *See also* Pair programming; Programming languages  
 acceptance tests and, 99–100  
 Bill of Rights for, 31  
 interviews with, 100–101  
 “picking the brains” of, 100  
 release planning and, 42–43  
 support of testers by, 229

Programming languages  
 compatibility of, with tests, 111–112, 157  
 selecting, 110

Project managers. *See* Coaches; Leadership; Teams

Projects, large, 216, 236, 237, 238–240

PUT method, 170

## Q

QA (quality assurance). *See also* Quality  
 definition of, 185  
 problems, XP as the solution to, 6–8  
 QC and, distinction between, 12  
 skills, importance of, for testers, 11–12  
 staff, introducing XP to, 226  
 user expectations and, 8

QC (quality control), 12. *See also* Quality

Quality. *See also* QA (quality assurance); QC (quality control)  
 acceptance tests and, 93–97  
 as a buzzword, 95  
 as collaborative effort, 38  
 as a core element of XP, 3  
 criteria, setting, 37

definition of, 35–37, 246–247  
 external, 95–96, 267  
 focus on, by testers, 250  
 internal, 95–96, 267  
 responsibility for, 37–38  
 striving for the highest standards of, 96  
 tools related to, 209

## R

Rebooting, 101–102

Records, locking, 36

Refactoring  
 acceptance tests and, 113  
 application test classes, 162–163  
 basic description of, 125–130  
 the direct-class interface, 161–162  
 grade cards and, 186  
 missing tests and, 195  
 quality assurance and, 37  
 road-trip metaphor and, 40  
 search test and, 129  
 simplicity and, 5  
 system state and, 127  
 testability and, 206  
 test infrastructure tasks and, 84  
 unit tests and, 120

Regression tests  
 basic description of, 194–195  
 iteration planning and, 79

Release(s). *See also* Release planning  
 basic description of, 196–197  
 dates, slipped, 196  
 deadlines, anxiety over, 196  
 small, 5, 8  
 which require several passes through the testing cycle, 225  
 worst-case scenarios regarding, 197

Release planning. *See also* Releases;  
Planning  
acceptance tests and, 57, 61, 62  
basic description of, 41–45  
enabling accurate estimates during,  
71–75  
hidden assumptions and, 51  
iteration planning and, 78  
road-trip metaphor and, 40  
role of testers in, 42–43  
widening activity during, 77

Remodeling projects, 12–14, 18, 30,  
36, 43–44, 51–52

Reporting, 86, 179–180, 209

Repository  
acceptance tests and, 19, 119  
unit testing and, 18

Requirements  
documentation, 238–239, 242  
gathering, 242–243  
minimum, 27  
missing, 7  
out-of-date, 7, 8  
retrospectives and, 190, 191  
satisfying, 238–239

Reset step  
automating, 126  
cost of, 126  
system state and, 126–128

Response time, reasonable, 94

Retesting, time estimates for, 44

Retrospective meetings, 185–192,  
209, 245–246, 284–285

Reverse-engineering, 141

Risk  
lights-out test design and, 105–106  
minimizing, 105

Roles, of testers, 14–15, 17–38,  
42–43, 78–81

Running tests, 147–146, 150–152,  
177–174  
overview of, 244–245  
time estimates for, 44  
special considerations for, 65, 67

## S

Sad path, 101, 103–104, 107–108,  
268, 269

Save button, 166

Schalliol, Gregory, 235, 236,  
237–238

Schedules, 196, 231–232, 238

Scripts. *See also* Scripts (listed by  
name)  
automation and, 140  
captured, 140–141, 209  
CGI (Common Gateway Interface)  
scripts, 159  
data-independent tests and,  
142–143  
modular tests and, 141–142  
self-verifying, 170  
shell, 89

Scripts (listed by name). *See also*  
Scripts  
AllTests.java, 154–155  
LoginStoryTest.java, 120, 122, 151  
SubmitRequest, 204–205  
TaskStoryTest.java, 279  
TestLoginStory.java, 121  
UserIDStoryTest.java, 152–153  
XTRACK, 165, 166  
XTrackDirectInterface.java,  
161–162  
XTRACKIF, 165, 166, 171, 176,  
282–284  
XTrackTest.java, 149, 153,  
162–163, 279–280

search() method, 112, 143  
 search module, 142–143  
 Search screens, 102–104  
 Self-verifying tests, 143–144  
 Servers, 96, 159, 246  
 setInterface() method, 168, 281  
 setTestInterface() method, 163,  
     165–166  
 Shell scripts, 89  
 Simplicity, 4–6, 25, 186  
 Sims, John, 198  
 “Smoke” testing, 244  
 Source code control, 79, 189, 204,  
     207, 209, 213, 221, 222, 223  
 Spreadsheets, 121–123, 275–276  
     for metrics, 219–220  
     test documentation, 238–239  
 Standup meetings  
     grade cards and, 186  
     notes, 218, 219  
     restricting the time taken up by,  
         218  
     retrospectives and, 186, 188, 190,  
         191  
 “Steering the Car: Lessons Learned  
     from an Outsourced XP Project”  
     (Collins and Kini), 42  
 Stories. *See also* Story cards  
     acceptance tests and, 43–44, 55,  
         56–59, 61–69, 113–115  
     assumptions in, identifying, 43–44,  
         48  
     basic description of, 41–45  
     creation of, 41–45, 276–277  
     release planning and, 73  
     road-trip metaphor and, 40  
     selection of, educating the  
         customer about, 180  
 Story cards, 179–180, 209

Struts framework, 166, 203  
 Submit button, 101, 166  
 SubmitRequest script, 204–205  
 Success  
     celebrating, 217  
     of XP testers, supporting, 228–229  
 Sustainable pace, importance of, 6  
 Syntax rules, 110  
 System state  
     assumptions about, identifying,  
         130, 275–276  
     establishing, 125–126  
     left unchanged by tests, 126–128

## T

Tasks  
     categories of, 79–80  
     creating, 116–117, 259–267,  
         272–273, 278–280  
     deleting, 116–117, 121, 259–267,  
         272–273  
     estimating, 80–81, 83–92  
     identifying, 79–80  
     retrospectives and, 190, 191  
     updating, 116–117, 121, 259–267,  
         272–273  
 TaskStoryTest class, 122, 156  
 TaskStoryTest.java, 279  
 TaskStoryTest.xsl, 122  
 tdata, 172, 173  
 Teams  
     acceptance tests and, 99  
     celebrating success and, 185  
     defect management and, 182  
     insuring that XP practices are really  
         implemented by, 9  
     iteration planning and, 78, 80–81  
     organization of office space for,  
         215–216

*Teams, continued*

- retrospectives and, 190
  - role of testers on, 14–15, 17–23
  - separate test, 87–91
  - XP testing values and, 23–28
- Testability, 203–206, 243, 286
- designing for, 203–204
  - real-life example of, 204–205
  - test tools and, 208
- Test automation
- acceptance tests and, 64, 67, 84, 90–91, 133–137, 194, 208
  - accessories and, 216–218
  - baseline comparison method and, 142–144, 145
  - basic description of, 133–137, 139–145
  - capture/replay, 140, 170
  - data-independent tests and, 142–143
  - design spike, 86
  - direct-call testing and, 166–167
  - evolving tools and, 208
  - legacy systems and, 203
  - mastering, 133–134
  - metrics and, 219–220
  - modular tests and, 141–142
  - self-verifying tests and, 143–144
  - test infrastructure and, 86
  - time estimates for, 44
  - tools for, 211–212
  - unit testing and, 7, 9, 18, 245
- TestCase file, 149–150, 171
- testCreate worksheet, 122–123
- testDelete() method, 128
- Tester(s)
- activities, 32
  - additional duties for, 222
  - assumptions about, 43

- basic description of, 11–12
  - Beck on, 43
  - Bill of Rights, 31–32, 78–79
  - contribution of, 12–14
  - goals of, 41–42
  - introducing people to, 228–229
  - iteration planning and, 78–81
  - Murphy’s Law and, 14
  - recruiting, 232–233
  - release planning and, 42–43
  - roles of, 14–15, 17–38, 42–43, 78–81
  - rotating, 232
  - shortages of, 232–233
  - success of, supporting, 228–229
  - up-front activities and, 42–45
  - user stories and, 41–42
  - “Test everything” syndrome, 105–106
- Test-first coding, 6, 19
- acceptance tests and, 55–56
  - advantages of, 244–245
  - basic description of, 18
  - quality assurance and, 37–38
- testInf, 163, 165
- Test infrastructure
- APIs and, 84
  - functional tests and, 85–87
  - iteration planning and, 80
  - tasks, defining/estimating, 83–92
- Testing.com Web site, 210
- testLogin() method, 122
- TestLoginStory class, 121
- TestLoginStory.java, 121
- testLogin worksheet, 122
- Test plans, 238–239, 243–244
- TestRunner framework, 151
- Test suites, combining multiple tests into, 154–155

**Test tools**

- basic description of, 169–176, 208–209
- cost of, 211
- evaluation copies of, 210–211
- evolving, 208
- experimenting with, 211–213
- implementing, 207–213
- interfacing to, 159–160
- off-the-shelf, 210–211
- selecting, 207–213
- testUpdate worksheet, 122
- Timestamps, 116, 270
- Tools. *See also* Test tools
  - building, 208, 213
  - implementing, 207–208, 211
  - selecting, 207–208, 210–211, 213
- Transactions, concurrent, support for, 36. *See also* Concurrent users

**U**

UML (Unified Modeling Language), 218

**Unit tests**

- acceptance tests and, 19–21, 55–56
- advantages of, 244–245
- automation of, 7, 9, 18, 245
- check and balances and, 18–19
- creation of, before generating code, 6
- effective, 245
- feedback and, 5, 26
- grade cards and, 186
- lack of adequate, 8
- metrics and, 220
- poorly written, problems caused by, 229
- quality assurance and, 96
- road-trip metaphor and, 40

rules for, 120

updateTask module, 266–267

Usability testing, 242

Usenet newsgroups, 210

User groups, 210

User IDs, 111, 113–115

creating, 126–128

deleting, 126–128

executable tests and, 150, 152–153

login module and, 172

UserIDStoryTest class, 154–156, 279, 284

UserIDStoryTest.java, 152–153

User interface. *See also* GUI (graphical user interface)

direct-call test automation and, 166

testing, 135–136

testing below, 167

Users, concurrent

acceptance tests and, 62, 67

hidden assumptions and, 52

number of, 52, 96

quality assurance and, 52, 96

User stories. *See also* Story cards

acceptance tests and, 43–44, 55,

56–59, 61–69, 113–115

assumptions in, identifying, 43–44, 48

basic description of, 41–45

creation of, 41–45, 276–277

release planning and, 73

road-trip metaphor and, 40

selection of, educating the customer about, 180

**V****Validation**

criteria, 173–175

direct-call test automation and, 166

Values. *See also* Communication  
 basic description of, 4, 23–28  
 courage, 4, 6, 25, 27–28, 30–31  
 feedback, 4–5, 26–29, 219–220  
 242, 244  
 simplicity, 4–6, 25, 186  
 vCrit, 172  
 Velocity, estimates, 44, 85, 274–275  
 verifyList module, 90–91  
 Version control, for acceptance tests,  
 119–120  
 vLevel, 172  
 VNC, 239, 240

**W**

Waterfall development, 106, 230, 241  
 WebART test tool, 160, 162–165  
 basic description of, 170  
 main script for, 170–171  
 retrospectives and, 188  
 WebARTPersistentScript object, 165  
 Webcams, 239  
 Web sites (listed by name)  
 JUnit Web site, 110, 111  
 Pettichord.com Web site, 210  
 Merriam-Webster Web site, 24  
 Pettichord.com Web site, 210  
 Testing.com Web site, 210  
 XPTester.org Web site, 194, 219,  
 238  
 Whiteboard, 198, 209  
 basic description of, 216–217  
 used for large or multilocation  
 projects, 239  
 Widening activity, use of the term, 77,  
 82, 265  
 Wiki, 186, 217, 218, 220  
 Wings-fall-off button, 135–136

Worst-case scenarios, 101–102, 197

**X**

XPTester.org Web site, 194, 219, 238  
 XP Universe 2001, 185  
 XTrack, 53–54, 60, 67, 107–108,  
 116–117, 255–259  
 automation and, 145  
 executable tests and, 148–156  
 expanding tests and, 159  
 interfacing to test tools and, 160  
 login modules and, 171–172  
 manual tests and, 137  
 quality assurance and, 95  
 retrospectives and, 192  
 running acceptance tests for, 183  
 spreadsheets and, 124  
 system state and, 130  
 task estimates for, 92  
 testability and, 206, 286  
 three major purposes of, 270–271  
 validation criteria and, 173–175  
 XTrackDirectInterface class,  
 161–162  
 XTrackDirectInterface.java, 161–162  
 XTRACKIF script, 165, 166, 171,  
 176, 282–284  
 XTRACK script, 165, 166  
 XTrackSession() method, 150  
 XTrackTest class, 148–150, 152–154,  
 158–168, 279, 281  
 XTrackTestInterface class, 160–161,  
 165  
 XTrackTest.java, 149, 153, 162–163,  
 279–280  
 XTrackWebARTInterface class, 163,  
 164–165, 171  
 XUnit, 110, 245