

# Index

## A

- Abundant living, 167
- Accepted responsibility, 4, 165
- Accomplishment, as human need, 24
- Accountability
  - community and, 158
  - executive role and, 78
- Accounting, for expense vs. investment, 113
- Accreditation, XP, 146–147
- Action, reflection following, 30
- Adopting XP. *See* XP, applying
- Alexander, Christopher, 153–154
- Analysis, decision making, 172
- Andres-Beck, Beth, 104
- Anxiety, accompanying change, 57
- Application development. *See* Software development
- Architects, team roles, 75–76
- Architecture
  - design and, 154–155
  - fluidity, 128
  - tests and, 75–76
- Architecture, of buildings, 162, 163
- Artifacts, of development, 66–67
- Attitude, bibliographic references, 162–163
- Auditing, projects prior to release, 116
- Authority
  - misalignment of authority and responsibility, 141
- Automated builds, 49
- Automated tests, 100–101, 171
- awareness, of need for change, 56–57

## B

- Baby steps, 33, 53
- Belonging
  - human needs, 24
  - team approach and, 39
- Beta testing, 101
- Bibliography, 161–174
  - attitudes, 162–163
  - emergent processes, 163–164
  - people, 165–168
  - philosophy, 161–162
  - programming, 171–174
  - project management, 168–171
  - systems, 164–165
- Big bang integration, 30, 87
- Big deployments, 63
- Biology, in 21st Century, 155

- Boehm, Barry, 52
- Bottlenecks
- coach noticing, 143
  - identifying, 47, 86–87
  - Theory of Constraints and, 85–86
- Brand, Stewart, 104, 174
- Breaks, in work day, 41–42
- Budgets, 94–95
- Business
- business interests dominating
    - development, 154
  - business interests sharing responsibility with programmers, 155
  - paradigm shifts and, 166
  - relationships, 1
- C**
- Capability Maturity Model, 150
- Capital expenditures, 113
- Certification, XP, 146–147
- Change
- accountability and, 158
  - adapting to, 11
  - awareness of need for, 56–57
  - baby steps and, 33
  - changing one thing at a time, 55
  - costs of, 52
  - deciding what to change first, 56
  - factors in rapid change, 142
  - feedback and, 19
  - opportunities for, 30–31
  - people and, 155
  - speed of, 56
  - starting with yourself, 57
  - strategies for, 168
- Chaos theory, 164
- Charts, in Informative Workspace, 41
- Chrysler Smalltalk project, 125–129
- estimation, 127–128
  - incremental design, 127
  - success of, 128–129
  - team creation, 126–127
  - trouble indicators, 126
- Clarity, bibliographic reference, 161
- Coach, selecting, 143–144
- Code
- code and tests, 66–67, 101–102
  - communicating through, 171
  - defect levels and, 98
  - eliminating duplication of, 108
  - future users, 26
  - as key in software development, xix
  - profitability of, 173
  - sharing responsibility for, 66
  - single code base vs. multiple code streams, 67–68
  - team approach to, 17
  - test-first programming and, 50
  - traceability of changes to, 116–117
  - trust and, 51
  - waste and, 137
- Code Complete* (McConnell), 104
- Coe, Bob, 126
- Cohesion, of code, 50
- Collective ownership, 66. *See also* Responsibility
- Comics, 166
- Commitment, waste created by over-commitment, 48
- Communication
- between business and technical people, 172
  - courage and, 21
  - credibility and, 48
  - documentation and, 146
  - drawings as, 174
  - embracing as a value, 146
  - feedback and, 20
  - listening skills vs. talking skills, 157
  - multi-site development and, 149
  - nonviolent, 167

- product managers encouraging, 78
  - programming as form of, 173
  - project managers responsibility for, 76–77
  - simplicity and, 19
    - as value guiding development, 18
  - Community, XP, 157–160
  - Computing, in 21st Century, 155
  - Conflict
    - community and, 158
    - diversity and, 29
  - Conquer-and-divide, 112
  - Consensus, in project management, 170
  - Constraints. *See* Theory of Constraints
  - Continuous improvement, 141–142
  - Continuous integration
    - collective ownership and, 66
    - as primary practice, 49–50
  - Contracts, ongoing negotiation of scope, 69
  - Contributing to Eclipse* (Gamma), 51
  - Control
    - fallacy of working longer to regain, 41
    - illusion of being able to control others, xxii
    - of people, 166
    - quality and, 32, 169
    - scope as basis of, 33
  - Cooperation, 18, 93
  - Costs
    - changes, 52
    - code development, 173
    - defects, 97
    - finding defects early and, 99
    - options pricing, 174
    - project management and, 92
    - redundancy, 31
    - software development, 173
    - variable in zero-sum model, 161–162
  - Coupling, of code, 50–51
  - Courage
    - balancing with other values, 21
    - executive role and, 78
    - multi-site development and, 149
    - as value guiding development, 20–21
  - Credibility, 48
  - Customers
    - development artifacts of value to, 66–67
    - driving system content, 12
    - evolutionary delivery and, 169
    - features controlled by, 128
    - interaction designers working with, 75
    - involvement of, xvi, 61–62
    - technical writers and, 80
    - Whole Team practice and, 39
- D**
- Daily deployment, xvi, 68–69, 143
  - Daily focus, of incremental design, 103
  - Database design strategy, 107–108, 172
  - DCI (Defect Cost Increase), 98–99
  - Deadlines, business concerns dominating, 154
  - Decision making
    - analysis decisions, 172
    - design decision, 172
    - in difficult situations, 165
  - Defect Cost Increase (DCI), 98–99
  - Defects, 119–121
    - acceptable levels of, 97–98
    - defect rate in Smalltalk project, 128
    - incremental design and, 52

- Defects, *continued*  
 metrics for defects after  
   deployment, 79  
   redundancy and, 31  
   root cause analysis, 64–65  
   tests for reducing rate of, 5  
   values and, 14
- Deming, W. Edwards, 167
- Deployment  
 daily, 68–69, 143  
 incremental approach to, 62–63  
 incremental design and, 109  
 metrics for defects after, 79
- Design. *See also* Incremental design  
 Alexander’s principles, 162  
 common language for decision  
   making, 172  
 database design strategy, 107–108,  
   172  
 patterns and, 108, 173  
 small scale, 171
- Developers. *See* Programmers
- Development. *See* Software  
 development
- Disney, 163
- Diversity principle, 29
- Documentation  
 code and tests as basis of, 66  
 communication and, 146  
 “Rosetta Stone” document,  
   114–115  
 technical publications, 80–81  
 of tests, 26  
 Unified Process document driven  
   basis, 169
- Double-checking, defect testing,  
 98–100
- Drawings. *See* Images
- Drawings, as communication  
 medium, 174
- DSDM (Dynamic Systems Develop-  
 ment Method), 170
- Dynamic Systems Development  
 Method (DSDM), 170
- E**
- Eclipse project, xv–xvi
- Economics  
 principles in XP, 25  
 quality and, 33
- Ego, thinking and, 165
- Emergent processes, bibliographic  
 references, 163–164
- Emotions, fear as barrier to perfor-  
 mance, 167
- Employees. *See* Staffing
- Energized work  
 map of, 58  
 as primary practice, 41–42
- Ernst, Michael, 51, 173
- Estimation  
 benefit of early estimation, 44–45  
 creating believable estimates,  
   127–128  
 planning and, 92, 93–94  
 real time estimates, 168  
 values and, 14
- Execution, separating from planning  
 in social engineering, 132
- Executive, as team role, 78–79
- Executive sponsorship  
 crucial to success of XP, 90,  
   119–121  
 finding, 140
- Expenses. *See* Costs
- Experience, design process and, 107  
 “An Experimental Evaluation of  
 Continuous Testing During  
 Development” (Saff and Ernst),  
 51, 173

**F**Facilities. *See* Workspace

Failure

dealing with consequences of,  
116–117

learning from, 143

as principle in XP, 32

Features

customer control of, 128

tracking projects by, 169

Feedback

from continuous testing, 173

Eclipse project and, xv

finding defects and, 99

measuring software projects, 169

pay-per-use, 69–70

reflection combined with doing,  
30

types of, 20

as value guiding development,  
19–20

Flow

principles in XP, 30

team approach and, 73–74

*The Forest People* (Turnbull), 4

Fowler, Martin, 95, 126

Fractals, 27

**G**

Gamma, Erich, 51

Gantt, Henry, 131

Gilbreth, Frank, 131

Gilbreth, Lillian, 131

Gladwell, Malcolm, 39

Global software development, 151

Goals

executive role and, 78

planning and, 91

XP goals for software develop-  
ment, xxiGraphics. *See* Images

Group dynamics, 143

Growth, as human need, 24

**H**

Health, pair programming and, 43

Hendrickson, Chet, 128

High-cost base areas, compared with

low-cost base areas, 150

Hiring, 81–82

History, practice of, 167

Hopelessness, overcoming, 163

*How Buildings Learn* (Brand), 104

Human resources, reviews and hir-

ing, 81–82

Humanity

fear as barrier to performance, 167

principle in XP, 24–25

Sit Together practice and, 38

workspace and, 40

Hunt, Andy, 140

Hygiene, 43

**I**Illnesses. *See* Sickesses

Images

communicating with drawings,  
174communicating with graphs and  
pictures, 174

Improvement

executive role and, 78

noncontinuous nature of, 142

principles in XP, 28

Incremental deployment, 62–63,

169

Incremental design, 103–110

daily focus of, 103

database design strategy, 107–108

deciding when to design, 105–107

Incremental design, *continued*

- Eclipse project and, xvi
- improvement as focus of, 28
- investing in, 172
- Once and Only Once heuristic, 108
- as primary practice, 51–53
- simplicity of design, 109–110
- Smalltalk project, 127
- timing of design decisions, 109
- weakness of physical-based metaphors for, 103–104

Industrial engineering, 131

Informative workplace, 39–41

- charts, 41
- human needs and, 40–41
- story cards, 40

Insight, 41

Integration, continuous integration practice, 49–50

Integrity, 159

Interaction designers, as team role, 75

Investments

- measuring investment-to-return, 79
- XP as expense or investment, 113

Iterations

- feedback cycles and, 7, 94
- planning frequency of, 121
- removing constraints or limitations, 168
- story implementation and, 127

**J**

JAD (Joint Application Development), 171

Jeffries, Ron, 126

Jensen, Brad, 119–121

Jobs, offshore development and, 150

Joint Application Development (JAD), 171

Judgement, communication and, 168

JUnit, xiv, 171, 173

Just In Time Software process, xiii–xiv

**L**

Leadership, 143

Learning

- applying new skills, 141
- conflict and disagreement and, 158
- by example, 143
- from failures, 32, 143
- reflection as basis of, 30

Life cycle models, 116

Listening skill

- community and, 157
- listening to feedback, 80, 141
- planning and, 93

Load tests, 101

Low-cost base areas, compared with high-cost base areas, 150

**M**

Maintenance

- applying XP to, 170
- project management and, 170

Management

- executives, 78–79
- product managers, 78
- project managers, 76–77, 92, 113–114
- Scientific Management and, 131
- self-organizing systems as metaphor for management, 164

Manual testing, 101

Manuals, 80–81. *See also*

Documentation

Margins, in software development, 165

- Mathematics, programming as, 172  
 McConnell, Steve, 104–105, 173  
 Meetings, weekly cycles, 46  
 Metaphors  
   chosen by interaction designers, 75  
   code names and, 26  
   driving XP, 12  
   physical-based impose limits on  
     software development, 104  
   Scientific Management, 131  
   self-organizing systems as meta-  
     phor for management, 164  
   thinking and, 162  
   Unified Process emphasis on, 170  
 Metrics  
   awareness and, 56  
   feedback and, 169  
   graphing, 174  
   for health of XP team, 79  
   measuring progress with tests, 102  
   for XP, 145  
 Micro-optimization, 88  
 Mistakes. *See* Failure  
 Modernism, 161  
 Money. *See also* Costs  
   pay-per-use and, 69–70  
   time value of, 25  
*The Mountain People* (Turnbull), 4  
 Multi-site development, 149–152  
   global software development, 151  
   high-cost base areas compared  
     with low-cost base areas, 150  
   practices and, 150  
   principles and, 150  
   reasons for, 149  
   values and, 149  
 Mutual benefit, as principle in XP, 26
- N**  
 Names, coding style and, 26  
 Negotiated scope contract, 69
- O**  
 Offshore development. *See* Multi-site  
 development  
 Ohno, Taiichi, 136, 174  
 Once and Only Once, heuristic for  
   incremental design, 108  
 Online communities, XP, 158  
 Opportunity, as principle in XP,  
   30–31  
 Option value, of systems and teams,  
   25  
 Organizations  
   reducing team sizes, 150  
   reverting to old habits, 140  
   scaling XP and, 113–114  
 Overall throughput, vs. micro-opti-  
   mization, 88  
 Overproduction, as waste, 136  
 Overwork, holding back effort  
   through, 6  
 Ownership, collective, 66. *See also*  
   Responsibility
- P**  
 Pain, as factor in quick change, 142  
 Pair programming  
   benefits of, 42–43  
   continuous integration and, 50  
   personal space and, 43–44  
   as primary practice, 42–43  
   reasons for applying, 35  
   teamwork and, 66  
   technical collaboration and, 57  
   XP building on, xiv  
 Paradigms, 166  
 Partitioning systems  
   architect’s responsibility for, 76  
   scaling XP and, 112  
 Patterns  
   design process and, 108, 173  
   XP and, xiv

- Pay-per-release, 70
- Pay-per-use, 69–70
- People
  - bibliographic references, 165–168
  - change and, 155
  - communication between business and technical people, 172
  - as component of problems, 38
  - scaling XP and, 111–112
- Perfection, 28
- Performance, fear as barrier to, 167
- Performance tuning, 93, 125
- Permaculture, 103, 162
- Personal space, 43–44
- Philosophy
  - bibliographic references, 161–162
  - of XP, 123
- Physical environment. *See* Workspace
- Planning, 91–95
  - Chrysler Smalltalk project, 127
  - deciding what to change first, 56
  - estimation and, 92, 93–95
  - goals and, 91
  - incremental, xvi
  - project managers responsibility for, 77
  - quarterly cycles and, 47–48
  - scope as basis of, 92
  - separating from execution in Taylorism, 132
  - team cooperation in, 93
  - technical details of, 168
  - timescales and, 92–93
  - weekly cycles and, 46–47
- Politics, of offshore development, 150
- Postmodernism, 161
- Practices
  - based on values, 14
  - compared with values, 14–15
  - defined, 13
  - implementing primary before corollary, 61
  - ineffectiveness of dictating, 57
  - learning by example, 143
  - mapping, 58–59
  - multi-site development and, 150
  - overview of, 35–36
  - social relationships and, 154
  - win-win-win, 26
- Practices, corollary, 61–73
  - code and tests, 66–67
  - customer involvement, 61–62
  - daily deployment, 68–69
  - incremental deployment, 62–63
  - negotiated scope contract, 69
  - pay-per-use, 69–70
  - root cause analysis, 64–66
  - shared code, 66
  - shrinking teams, 64
  - single code base, 67–68
  - team continuity, 63–64
- Practices, primary, 37–54
  - continuous integration, 49–50
  - energized work, 41–42
  - incremental design, 51–53
  - informative workplace, 39–41
  - pair programming, 42–43
  - quarterly cycles, 47–48
  - sit together, 37–38
  - slack, 48
  - stories, 44–45
  - ten-minute build, 49
  - test-first programming, 50–51
  - weekly cycles, 46–47
  - whole team approach, 38–39
- Predictability, as value, 22
- Principles, 23–36
  - baby steps, 33
  - defined, 15
  - diversity, 29
  - economics, 25



- failure, 32
  - flow, 30
  - humanity, 24–25
  - improvement, 28
  - learning by example, 143
  - multi-site development and, 150
  - mutual benefit, 26
  - opportunity, 30–31
  - overview of, 23
  - quality, 32–33
  - redundancy, 31–32
  - reflection, 29–30
  - responsibility, 34
  - self-similarity, 27–28
  - social relationships and, 154
  - Priorities, 109
    - aligning, 55–57
    - business, 67
    - economics of, 25
    - funding, 129
    - implementing highest priority first, 7–8
    - product managers and, 77
  - Problems
    - complexity in scaling XP, 115
    - as opportunity for change, 30–31
    - people-oriented solutions, 38
    - resolving in flow-based environment, 30
    - steps for working with big, 112
  - Product development, 170
  - Product managers, 77–78
  - Productivity
    - Energized Work principle and, 41
    - Scientific Management and, 131
    - TPS, 136
  - Programmers
    - global demand, 151
    - sharing responsibility with business interests, 155
    - as team role, 81
  - tests, 100
  - working with sponsors and users, 154
  - Programming
    - art of writing, 166
    - balancing human interests, 153–155
    - bibliographic references, 171–174
    - continuous integration practice, 49–50
    - pair programming principle, 42–43
    - for and by people, 168
    - pragmatic programmers, 140
    - short-cycle, 169
    - social and technical networks, 164
    - test-first programming, 50–51, 141, 143
  - Project management
    - bibliographic references, 168–171
    - Taylorist perspective, 170
  - Project managers
    - presenting information to organizations, 113–114
    - story cards and, 95
    - as team role, 76–77
  - Projects
    - cancellations, 5
    - feedback and, 169
    - tracking projects by features, 169
    - trouble indicators, 126
  - Pull, model of development, 87–88
  - Push, model of development, 87–88
- Q**
- Quality
    - principles in XP, 32–33
    - project management and, 92
    - quality control in Deming’s model, 167
    - social engineering and, 132–133
    - variable in zero-sum model, 161–162

- Quality-of-life, 22  
 Quarterly cycles, 47–48, 114
- R**
- Redundancy principle, 31–32  
 Refactoring, xiv, xv, 172  
 Reflection principle, 29–30  
 Regression testing, 65  
 Relationships  
   business relationships, 1  
   community, 157  
   fostering strong, 154  
   improving, 146  
   mutual benefit as basis of, 26  
   relational skills of programmers, 81  
   separating intimate relationships from work setting, 43  
   in societies of abundance and scarcity, 167  
   undermined by misalignment of authority and responsibility, 141  
 Release cycle, reducing, 6  
 Requirements  
   gathering, 137  
   misused terminology in development, 44  
 Resources, in societies of abundance and scarcity, 167  
 Respect  
   multi-site development and, 149  
   in Ohno's management approach, 174  
   as value guiding development, 21  
 Responsibility  
   accepted, 4, 165  
   vs. control by others, xxii  
   misalignment undermines trust, 141  
   as principle in XP, 34  
   shared code and, 66  
   sharing between programmers and business interests, 155  
 Revenue, measuring investment-to-return, 79  
 Review, of human resources, 81–82  
 Rewards, as control mechanism, 166  
 Risk  
   big deployments and, 63  
   daily deployment and, 68  
   economic, 26  
   of error, 49  
   of failure, leading to success, 32  
   management, 73, 116, 169  
   negotiated scope contract and, 69  
   not asking others to take risks you are not willing to take, 141  
   partitioning and, 112  
   silence as sound of risk piling up, 79  
   XP addresses at all levels, 7  
 Risk, in development process, 5–6  
 Roles flexibility, in XP programming, 82–83  
 Root cause analysis, 64–66  
 “Rosetta Stone” document, 114–115
- S**
- Sabre Airline Solutions, 119–121  
 Sadalage, Pramod, 107  
 Safety  
   human needs, 24  
   Sit Together practice and, 38  
   as value, 22  
 Saff, David, 51  
 Scaffolding, incremental deployment, 63  
 Scaling XP, 111–117  
   consequences of failure, 116–117  
   investments, 113  
   organization size, 113–114

- overview of, 111
- people, 111–112
- problem complexity and, 115
- solution complexity and, 115–116
- time, 114–115
- Schedules, slipping, 5
- Scientific Management, 131–132, 174
- Scope
  - business concerns dominating, 154
  - as control mechanism, 33
  - ongoing negotiation of, 69
  - planning as means of managing, 92
  - variable in zero-sum model, 161–162
- Scope creep, 50
- Seasons, as organizational timescale, 47
- Security
  - certifiable, 116–117
  - as value, 22
- Self-organizing systems, 164
- Self-similarity principle, 27–28
- Sexuality, in work environment, 43
- Shape, self-similarity principle, 27
- Shared code, 66
- Shrinking teams, 64
- Sicknesses, 41
- Simplicity
  - bibliographic reference, 161
  - courage and, 21
  - dealing with excess complexity, 115–116
  - feedback and, 20
  - incremental design, 109–110
  - multi-site development and, 149
  - as value guiding development, 18–19
- Single code base, 67–68, 150
- Sit together as a practice, 37–38, 145
- Skiing, 165
- Skills, learning and applying, 141
- Slack, as primary practice, 48
- Social change, 1
- Social engineering, 132–133
- Social relationships
  - stratification lacking in TPS, 136
  - XP applied in context of, 139
- Software development
  - advantages of XP for, 3–4
  - community for, 157–160
  - costs, 173
  - cycles, xvi
  - driving with stories, 169
  - DSDM approach to rapid development, 170
  - electrical engineering paradigm, 169
  - global, 151
  - goals of XP and, xxi
  - limitations of Taylor's model when applied to, 132
  - low-cost base areas vs. high-cost base areas, 150
  - margins in, 165
  - overproduction, 136–137
  - push model contrasted with pull model, 87–88
  - risk in, 5–6
  - shortcomings of Taylorist approach, 166
  - team-driven process, 12
  - Theory of Constraints and, 168
  - utility vs. technical virtuosity, 154
  - values guiding, 18
- Software engineering, 49
- Solution complexity, 115–116
- Sponsors
  - executive sponsorship, 90, 119–121, 140
  - working with developers and users, 154

- Staffing
    - managing turnover, 6
    - scaling, 112
    - needs of good developers, 24
    - worker responsibility in TPS, 135–136
  - Static verification, 101
  - Stories
    - breaking into tasks, 47
    - deciding what to change first, 56
    - driving development from, 169
    - interaction designers writing, 75
    - planning and, 91, 93–95
      - as primary practice, 44–45
    - product managers writing, 77–78
    - project completion time and, 127
    - slack time and, 48
    - weekly cycles and, 46
  - Story cards
    - example, 45
    - in informative workplace, 40
    - in planning process, 96
    - presenting information to organizations, 113–114
  - Stress tests, 101
  - Subscription model, software marketing, 70
  - Success
    - as goal, 146
    - XP and, 4
  - Survival, problem solving and, 31
  - Systems
    - bibliographic references, 164–165
    - self-organizing, 164
- T**
- Talking skills, 157
  - Tasks, breaking stories into, 47
  - Taylor, Frederick, 131–133, 150, 165, 167, 170
  - TDD (Test-Driven Development), 171
  - Team. *See also* Whole team practice
    - approach to coding style, 17
    - balancing individual needs with team needs, 24
    - certification and accreditation, 146
    - common factors in good software development teams, xxi–xxii
    - communication as basis of cooperation, 18
    - continuity, 63–64
    - Disney's, 163
    - diversity, 29
    - models, 66
    - orientation in XP, 6
    - reducing size (shrinking) of, 64
    - respect as key value to working of, 21
    - reverting to old habits, 140
    - scaling XP and, 112
    - sexuality complicating working of, 43
    - sharing power, 155
    - size thresholds, 39
    - software development as team-driven process, 12
    - things that can go wrong, 168
    - undermined by misalignment of authority and responsibility, 141
  - Team continuity, 63–64
  - Teamwork models, 66
  - Technical aspects
    - communication between business and technical people, 172
    - excellence in, 4
    - technical fixes must be complemented by people-oriented solutions, 38

- Technical collaboration, 57
- Technical employment, 150
- Technical publications, 80–81
- Technical writers, as team role, 80–81
- Technique, as basis of practices, 13
- Ten-minute build, as primary practice, 49
- Test-Driven Development (TDD), 171
- Test-first programming, 50–51, 141, 143, 171
- Testers, as team role, 74–75
- Tests, 97–102
  - automating, 100–101
  - code and test cycle, 66–67, 101–102
  - DCI, 98–99
  - defect rates, 5
  - defect reduction, 97–98
  - documenting, 26
  - double-checking, 100
  - early and often, xvi
  - feedback from continuous testing, 173
  - frequency of, 100
  - JUnit, 171
  - learning from failures, 32
  - measuring progress with, 102
  - regression testing, 65
  - static verification, 101
  - system architecture, 75–76
  - ten-minute build, 49
  - test-first programming, 50–51, 141, 143
  - unit tests, 173
  - weekly cycles and, 46, 74
- Theory of Constraints, 85–90
  - bottlenecks and, 85–86
  - identifying constraints, 86–87
  - overall throughput vs. micro-optimization, 88
  - push model of development contrasted with pull model, 87–88
  - software development and, 168
  - statement of theory, 86
  - understanding systems, 164
  - XP shifting constraints to non-software development areas, 89–90
- Thinking
  - ego and, 165
  - linear vs. nonlinear, 174
  - metaphors and, 162
- Thomas, Dave, 140
- ThoughtWorks, 107
- Throughput, 88, 164
- Time
  - long-running projects and, 114–115
  - planning and, 92–93
  - project management and, 92
  - quarterly cycles and, 47–48
  - seasons and, 47
  - time value of money, 25
  - variable in zero-sum model, 161–162
  - weekly cycles and, 46
- The Tipping Point* (Gladwell), 39
- Toyota Production System* (Ohno), 137
- Toyota Production System (TPS), 135–138
  - parallels to software development, 136–137
  - production process, 136
  - social stratification lacking in, 136
  - waste reduced, 135–136
  - worker responsibility in, 135–136
- Tracking, projects by features, 169

Trust  
 defects and, 97–98  
 undermined by misalignment of  
 responsibility, 141  
 Turnbull, Colin, 4

## U

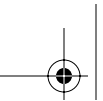
Underwork, holding back effort  
 through, 6  
 Unit tests, xiv, xv, 173  
 UP (Unified Process), 170  
 User-interface design, 166  
 Users. *See also* Customers  
 as team role, 81  
 technical writers and, 80  
 Users, *continued*  
 tests based on perspective of, 102  
 working with developers and spon-  
 sors, 154

## V

Values, 17–22  
 based on what really matters, 17  
 change and, 56  
 communication, 18  
 compared with practices, 14–15  
 courage, 20–21  
 defined, 14  
 feedback, 19–20  
 guiding development, 18  
 improvement and, 142  
 integrity and, 159  
 learning by example, 143  
 multi-site development and, 149  
 not using XP when organization  
 values at odds with XP values,  
 144  
 other important, 22  
 respect, 21  
 simplicity, 18–19

## W

Wabi-Sabi, 161  
 Waste  
 customer involvement in reduc-  
 ing, 61  
 eliminating, 28  
 overcommitment and, 48  
 overproduction and, 136–137  
 planning as necessary waste, 46–47  
 redundancy and, 32  
 Toyota success in eliminating,  
 135–136  
 Waterfall process, 87, 146  
 Weekly cycles, 46–47, 74  
 Whole team practice, 73–83  
 architects, 75–76  
 customers, 61–62  
 executives, 78–79  
 failure to work together,  
 73–74  
 human resources, 81–82  
 interaction designers, 75  
 overview of, 38–39  
 product managers, 77–78  
 programmers, 81  
 project managers, 76–77  
 role flexibility and, 82–83  
 technical writers, 80–81  
 testers, 74–75  
 users, 81  
 Win-win-win practices, 26  
 Work hours  
 balancing with other human needs,  
 24  
 energized work principle and, 41  
 Workspace  
 design of, 163–164  
 informative workspace practice,  
 39–41  
 sit together practice, 38



## **X**

XP, applying, 139–144

coach selection, 143–144

executive sponsorship, 119–121,  
140

improvements, 142

learning and applying skills, 141

organization reverting to old hab-  
its, ways of doing things, 140

social relationships and, 139

staring with yourself, 140–141

when not to apply XP, 144

XP, getting started, 55–59

awareness of need for change,  
56–57

change starts with yourself, 57

changing one thing at a time, 55

deciding what to change first, 56

mapping practices and, 58–59

XP, overview

aspects of, 2

benefits of, 3

business relationships and, 1

certification and accreditation,  
146–147

constraints shifted to non-  
software development areas,  
89–90

defined, iv, 6–7

distinguishing characteristics, 2

metrics for, 145–146

risk in development process and,  
5–6

social change and, 1

success and, 4

## **Z**

Zero-sum model, 161–162