

INDEX

- Abandoned software project, 217
- Abdel-Hamid, T., 188, 195
- Accounts receivable, 253
- Accused project managers, 221
- ACNielsen, 118, 120, 122
- Active users, 70, 72, 76
- Ad hoc, 288
- Ad hoc environment, 283
- Ad hoc programming tasks, 259
- Adaptive, 194
- Adaptive models, 6
- Adding manpower, 188, 190
- Additive tasks, 149, 151
- Agile manifesto, 60, 83, 212, 217–218
- Agile methods, 14
- Agile practices, 10, 33–34, 41, 59, 78, 82–83, 145, 212, 241, 283
- Agile project management, 145, 237
- Agile project progress, 229
- Agile software development (ASD), 10, 57, 60–61, 63–64, 73, 77, 83, 150, 156, 163, 199, 224, 227, 240–241
- Agile software development processes, 34
- Agile team, 212
- Agile teaming, 197–200, 202–204, 206, 208, 210, 212–216, 218–220, 222, 224
- Agile teaming rhythm, 215
- Agile values, 63–64, 78, 83
- Agreed-on plans, 4
- Alexander, C., 100, 262, 263
- Algorithms, 40, 176
- ALICE, 124, 126
- All-the-time single pair programming, 180
- Among-classes, 256–257
- Analysis and design as planning, 5
- Analysis-design process, 5
- Analyst programmers, 223
- Andres, C., 24, 34, 59, 199, 215, 224
- ANNA, 126
- Antigroup phenomenon, 145, 168
- Anton, J., 190, 195
- Application, 12–13, 15–17, 88–89, 101–102, 108, 110, 112, 151–152, 180, 220, 247, 255, 266, 271–273
- Application requirements, 66, 115, 174
- Applying development rhythms, 281, 288
- Architecture, 100, 106
- Arrhythmic pairing, 205
- Arrhythmic, 204, 206
- Arrhythmic software project, 213
- A/S400, 220, 222
- ASD, *see* Agile software development
- Asia, 46
- Assembly line, 151–152, 243–246
- Assembly task planning, 7, 8
- Asynchronously collaborating, 56
- ATM, 154
- Atomic Energy of Canada Limited, 4
- Atomic tasks, 142
- A-Tutor, 68, 116–117
- Automated test cases, 81, 277–278
- Automated unit tests, 79, 278, 286
- Awkward team structure, 23

- Backward pass, 143
- Bad-smell concept, 258
- Bank accounts, 154
- Bank, 154
- Basic project management techniques, 240

- BASIC, *see* Beginners' All-Purpose Symbolic Instruction Code
- Bazaar models, 65
- Beck, K., 24, 34, 59, 199, 215, 224, 249, 263, 267, 274, 277, 288
- Beckham, David, 121, 125
- Beedle, M., 161, 163, 203, 224
- Beginners' All-Purpose Symbolic Instruction Code (BASIC), 89
- Bernstein, L., 21, 35
- Big-bang implementation, 23
- Bill of Materials (BOM), 7–9
- Black, J. T., 240, 263
- Blackbox programming, 74, 76, 78
- Blackbox testing, 108
- Blackburn, J. D., 249, 263
- Boehm, B., 15, 35
- BOM, *see* Bill of Materials
- Bonus pay, 18
- Bottom-up approach, 230
- Boundary values, 272
- Breakpoint testing, 271, 275
- Brooks' law, 14, 188, 190–191, 194
- Brooks, F. P., 35, 188, 195, 267, 288
- BS5750, 4
- Budget, 12
- Budget control, 188
- Business continual process improvement, 14
- Business logic, 156
- Business operations, 28, 40–41, 116, 225, 232
- Business processes, 22, 205
- C, 12, 90, 125, 261
- C#, 283
- Calls for a partner exchange, 159
- Capability maturity model, *see* CMM
- Capability maturity model integrated, *see* CMMI
- Capretz, L. F., 43, 54
- Carr, N. G., 131, 162
- CASE, *see* Computer-aided software engineering
- Cathedral models, 65–66, 70, 75
- Causal relationships, 51
- Challenging programming problems, 176
- Chan, K. C. C., 176, 195
- Changing requirements, 23, 223, 253
- Chaos models, 21, 35
- CHAOS report, 253
- Chapman, S. N., 8, 35
- Chatting robot, 122
- Chief programmers, 198
- China, 4, 45, 47, 90, 96, 119, 122–123, 187, 220, 241, 262, 292
- Chinese marketplace, 90
- Chords, 291
- Chou, H. W., 198, 205, 224
- Class component, 8
- Class diagrams, 8–10, 65
- Class hierarchy, 254
- Class inheritance, 156
- Class relationships, 257, 259, 262
- Class-relationship-restructured, 256–257
- Clients, 21, 65, 67, 73, 108, 142–144, 235, 261
- Clipper, 12, 225
- Closed-source software project, 66, 69
- CMM, 4, 240–242, 283, 288
- CMM levels, 240
- CMM rhythm, 240–241
- CMMI, 4, 198–199, 240, 283, 288
- Coach software team, 28
- Coaching pair programming teams, 161
- Coad, P., 93, 129
- Cockburn, A., 81, 83, 138, 162, 195
- Code comprehension, 39
- Code patterns, 12, 293
- Code quality, 67, 76, 271, 280
- Code readability, 156
- Code samples, 94, 102, 108, 110–113, 115, 126
- Code-and-fix, 16–17, 21
- Code-driven refactoring, 259
- Code-oriented, 102
- Coders, 19
- Codevelopers, 75–76
- Coding, 16–17, 27, 63–64, 87–90, 92–96, 105–108, 131–133, 135–139, 156–161, 233–238, 261–262, 266–269, 271–275, 277–279
- Collaboration, 10, 41, 57, 65, 67, 69, 71–72, 75, 135–137, 169, 171, 184, 217–218, 238
- Collaborative cognition, 132
- Collaborative programming, 81, 132, 163, 195
- Collaborative work, 133, 210
- Collective code ownership, 187, 210
- Collective tasks, 209, 211
- Collocated team, 78–79, 82
- Commercial products, 68–69, 74
- Commercial projects, 71, 286
- Commercial software project, 66, 71, 81
- Communication channels, 111, 169, 192–193
- Communication costs, 188, 191–193, 233–234
- Communication overheads, 192
- Communication proximity, 79, 80

- Complementary skills, 203
- Completeness of source code, 73–74
- Completion time, 17–18, 20, 133, 143, 176, 185, 188, 192, 274
- Computer languages, 107, 261
- Computer programming, 39, 137, 168
- Computer-aided software engineering (CASE), 154
- Concurrent software, 20, 263
- Concurrent software engineering, 20
- Conflicting requirements, 251, 253
- Conflicts, 161
- Constant feedback, 6
- Constantine, L. L., 133, 162, 195
- Contemporary software design, 231
- Continuous improvement, 240
- Continuous integration, 79, 81, 152, 161
- Control projects, 12
- Cooperative learning, 137
- Copy-and-paste programming, 88, 128
- Copyleft, 117
- Core developers, 59, 70, 75–76
- Core software programming, 59
- Cost of change, 249
- Costs, 4, 7, 48, 58, 119, 136, 140, 156, 161, 168, 170, 191–192, 233–234, 249
- CPE, 159
- CPM, *see* Critical path method
- CPX, 159
- Creativity, 134
- Crispin, L., 271, 288
- Critical path method (CPM), 142–143, 227
- CRM, *see* Customer relations management
- CRM applications, 110, 205, 253
- CRM project development cycle, 115
- CRM project, 111–112, 114
- CRUD, 95
- Cultural capital, 45–46
- Cultural differences, 201
- Cultural elements, 34
- Cultural understanding, 47
- Customer(s), 22
- Customer collaboration, 60
- Customer feedback, 12
- Customer relationships, 21
- Customer-relations management (CRM), 22, 40, 82, 110, 114–115, 145, 190, 230, 253. *See also* CRM entries
- Cut-and-paste programming, 88
- Data locking, 226
- Database, 10
- Database administration, 102
- Database denormalization, 102
- Database programming, 151
- Dave, 124
- Deadline, 281
- Deadlock handling, 226
- Deal or No Deal, 10
- Death march, 267
- Debugging, 69
- Defect(s), 9, 10, 73, 139, 152, 244–246, 249, 253, 262, 269–270, 272, 278, 286
- Defect detection, 139
- Delayed time, 20
- Deliberate mistakes, 275, 286
- Delivery lead time, 251
- Delphi, 17
- Deming, W. E., 14
- Demographic data, 118–120
- Departmental team, 201
- Design, 154, 168, 176, 268
- Design and testing, 152
- Design by code, 156–157, 159, 233, 268–269
- Design by diagram, 233
- Design defects, 5, 139, 159
- Design documents, 49, 50, 156
- Design flexibility, 102
- Design patterns, 101–102, 129, 185, 187, 261–263
- Design problems, 9, 105
- Design review, 106
- Design rhythm, 269
- Design software, 225
- Design solutions, 161, 259, 261–262, 273
- Developed areas, 47
- Developer role, 71–72
- Developers, 12, 16–18, 23–24, 50, 58–60, 62–65, 67–75, 78, 80–81, 102–104, 134, 189, 198–199, 281–283
- Developing software, 4, 5, 7, 9, 11, 175, 227, 241, 261
- Development environment, 31, 40, 81, 83, 171
- Development framework, 10, 25
- Development languages, 207
- Development methodology, 11, 24, 194, 283
- Development processes, 5, 20, 22, 29, 62
- Development rhythms, 25, 28, 30, 32, 34, 39, 42, 48, 53, 82–83, 158, 194, 275, 291–292
- Development team, 22, 29, 69, 78, 112, 114, 229, 268
- Devito Da Cunha, A., 43, 54
- Dietz-Uhler, B., 136, 163

- Discretionary tasks, 152, 203
- Distributed environment, 115
- Distributed pair programming, 80
- Distributed team, 78–79
- Diversity, 134
- Dividing and conquering, 5
- Documents, 57, 60, 148
- Domain knowledge, 22, 151, 207
- Double-loop iterative model, 14
- Dreamweaver, 47
- Dummy solutions, 275
- Duncan, S. D., 40, 54
- Dynamic, 194, 202
- Dynamic analysis, 270

- Easy to do, 99
- Easy to follow, 99
- Easy-to-start, 96, 158, 160, 238–239, 248
- Easy-to-sustain, 96, 158, 160, 238–239, 248
- Ebrahimi, A., 107, 129
- Economical number of test cases, 266–267
- Economics, 140
- Elapsed time, 228
- ELIZA, 124
- Empirical software, 175, 195
- Empirical software engineering, 175
- End-user, 71
- Engineering product defects, 5
- Engineering projects, 3, 7, 8
- English language, 46–47
- Enhanced waterfall, 6
- Enterprise resources planning (ERP), 71–72, 95, 97, 118, 149, 152, 177, 216, 229–230, 251
- Entity-relationship diagrams, 60
- Epley, N., 52, 54
- ERP, *see* Enterprise resources planning
- Error detection, 138
- E-training project, 116
- Eureka task, 150
- “Eureka”-type problem, 150
- Europe, 46
- Evaluating pair programming, 195
- Event-driven programming, 12, 90, 125
- Evolutionary software development, 14
- Evolution through prototyping, 247
- Ewusi-Mensha, K., 217, 224
- Exceptional handling, 23, 174
- Execution, 5
- Ex-partner exchange, 161
- Experienced management, 48–49, 51, 53
- Experienced programmers, 96, 135–136, 140, 159, 173, 253, 262, 265, 268–269
- Experimental evaluation of pair programming, 163, 195
- Experiments, 165
- Expert programmers, 140, 173
- Explicit knowledge, 31
- External behaviors, 254, 256, 258, 268, 275, 277
- External user interfaces, 271
- EXtreme programming, 8, 24–25, 34–35, 63, 78, 81–83, 114, 132, 152, 162–163, 175, 179, 194, 288–289

- Face-to-face meeting, 20
- Factors, cultural, 39
- FAQ, *see* Frequently asked questions
- Fast-paced, 194
- Fermi question, 150
- FIFO, *see* First-in first-out
- Final-year programming assignment, 88
- Finance director, 221
- Finished products, 14–15, 151, 244–245
- Firefox, 72
- First-in first-out (FIFO), 174, 254
- Flor, N. V., 155, 162, 195
- Flowcharts, 25, 155, 157, 243, 259, 270
- Forer, B. R., 51, 54
- Forgotten requirements, 251, 253
- Formal methods, 153
- Formation phases, 206
- Forsyth, D. R., 210, 224
- Forward pass, 143
- Four Ps, 61, 268
- Four-stage waterfall model, 5, 18
- Fowler, M., 258, 263
- FoxPro, 225
- Free riders, 211
- Freelance programmers, 270
- Frequently asked questions (FAQ), 132
- Function points, 202, 211, 227, 246, 266, 269
- Functional diversity, 205
- Functional modules, 212–213, 216–217, 222, 229
- Functionality enhancement, 69
- Functions, 13, 76, 110, 116, 128, 148, 151, 179, 209, 215, 230, 232, 246, 272–273
- Funded project, 57, 68

- Game theory, 28
- Game theory analysis, 29
- Gamma, E., 101, 129, 262, 263, 277
- Gang of four, 101
- Gardner, H., 173, 195
- Gartner Group, 22
- General public licence (GPL), 109

- Giddens, A., 198, 224
 Gifted programmers, 234, 237
 Global software team, 81
 GNU, 109
 GNU GPL, 109, 126
 Gödel, K., 137, 162
 GPL, *see* General public licence
 Grading, 258
 Graphical user interfaces (GUI), 179, 183–184, 186–187, 256, 259
 Greathead, D., 43, 54
 Group dynamics, 34, 134, 198, 224
 Group learning, 169
 Group of programmers, 41, 65, 134, 173
 GUI Creation Maintenance Inquiry, 183
 GUI, *see* Graphical user interfaces
 Gunther, R. E., 262, 264, 273, 289
- Halstead, M., 67, 83
 Hansen, J., 134, 163
 Hardcode development, 136
 Harrison, D. K., 239, 263
 Harrison, W., 246, 263
 Haslam, A., 165, 195
 Hawking, Stephen, 156
 Hawthorne effect, 166
 Heavyweight, 59, 60, 78, 82, 194
 Heavyweight processes, 59, 60
 Herzberg, F., 176, 195
 Heterogeneous team, 201
 High cohesion, 105
 Higher-quality software, 132
 Highly maintainable code, 236
 Hired programmers, 107, 135
 Hong Kong, 11, 116–117, 123–124, 200
 Horse-trading problem, 145, 150
 Human-centered, 38
 Human-computer studies, 173
 Human dynamism, 28
 Human programmers, 184, 276
 Human resources, 16–17
 Hunter, S. L., 240, 263
 Hutchins, E., 155, 162
 Hybrid approach, 230
 Hypnotic decision making, 51
- Iago, 222–223
 ICQ, 111, 122–126, 135
 IcqOscar] plug-in, 128
 Idle time, 20
 Immature prototype, 14
 Implementation, 9
 Improvisation, 237
 Incentive scheme, 18–19
- Incomplete requirements, 151, 253
 Incremental design, 79, 81, 225–234, 236–242, 244, 246–250, 252, 254, 256, 258, 260, 262, 264, 268
 Incremental development, 248
 India, 46
 Inexperienced programmers, 47, 107, 286
 Information technology, *see* IT entries
 Informix, 12
 In-out diagram, 95, 179, 239
 Inspectors, 168
 Integrate, 168
 Intentional beliefs, 52
 Invoice, 251
 ISO 9000, 4
 IT, 16–17, 131, 162, 201, 220, 222
 IT managers, 223
 IT projects, 216, 220
 Iteration cycles, 20–21
 Iterative model, 14, 17, 24, 29
 Iterative software development, 12
 Iterative software processes, 274
 Iterative waterfall model, 13–15, 25, 31
- Jacobson, L., 37, 54
 Janzen, D., 283, 288
 Java, 47
 Java programming, 150, 277
 Java programs, 43, 101, 154, 259
 Jespersen, 220–221
 JIT, *see* Just-in-time
 JIT software development, *see* Just-in-time software development
 Job(s), 17, 39, 43, 45, 48, 52, 97, 103–104, 122–123, 133–135, 137–138, 140, 167–168, 175
 Job simulation test, 96, 99
 Job test, 96–97
 Joone, 72
 Jorgensen, M., 51, 54
 JSPWiki, 72–73
 Junior programmer's code, 187
 JUnit, 277, 283
 Just-in-time (JIT), 159, 239–242, 245–247
 Just-In-time Software Development, 239, 241, 243, 245, 247
- Kahenman, D., 11
 Kameda, T., 152, 163
 Kanban, 159, 245
 Kaner, C., 113, 116, 130
 Keefer, G., 139, 163
 Keil, M., 219, 224
 Kessler, R., 133, 139, 163, 196, 211, 224

- Key performance indicator (KPI), 118–120
- Key users, 252–253
- Knowledge, 31, 34, 38, 40–41, 48, 63, 76, 81, 90–91, 116, 137–138, 203–204, 253, 255
- Knowledge of purpose, 261
- Knowledge of structure, 261
- Knowledge sharing, 169
- KPI, *see* Key performance indicator
- Labyrinthine pattern of software design, 231
- Lack of user input, 253
- Large project, 67, 78
- Last-in first-out (LIFO), 254
- Late software project, 213
- Late-project team, 193
- Lean manufacturing, 240
- Lean production, 239–240, 264
- Lean software development, 35, 240, 288
- Learning, 38, 45, 48–52, 54, 61, 130, 137, 153, 168, 208–209, 215, 280, 291
- Learning curve, 83, 104, 192, 206–209
- Learning speed, 208
- Level 3 refactoring, 258
- Leveson, N., 4, 35
- License, general-public, 109
- Lifecycle, 202, 207–208
- LIFO, *see* Last-in-first-out
- Lightweight processes, 58
- Lightweight, 59, 60, 62, 78, 82, 194
- Linearized Einstein equation, 156
- Linux, 57, 67–68, 84
- Linux's law, 145
- Load test, 174
- Local software team, 46
- Logical design, 5
- Logistics, 239
- Long meetings, 27
- Loosely coupled team, 80
- Luger, G., 155, 163
- Lui, K. M., 176, 195
- Madnick, S., 188, 195
- Maier, 145
- Maintain software, 154
- Maintainability, 235, 259–60
- Maintenance, 176
- Management, 10, 47, 77, 114, 120, 131, 136, 188, 190, 216–217
- Management support, 223
- Management theory, 175
- Managers, 11, 41, 43, 45–46, 48, 72, 82, 107, 110, 138, 200–201, 222–223, 231, 283–284
- Managing software project, 5, 60, 194, 241
- Managing software team, 47
- Manufacturing, 9, 46, 151, 159, 209, 239–240, 246–247, 263
- Manufacturing process, 242–243
- Manufacturing production, 3
- Manufacturing resource planning (MRP), 40
- Marketing research, 118–120
- Master and apprentices, 100
- Master-coach diagram, 31–33, 83, 101–102, 282, 292
- Matrix, cross-product, 95
- Maturity model, capability, 240
- Mayer, D. B., 39, 54
- McCabe, T., 67, 83
- McConnell, S., 105, 130
- Meeting, 24, 27–28, 38, 41, 49, 64, 106, 203, 220, 222–223, 252
- Mens, T., 234, 262, 264
- Message chains, 261
- Messaging unit, 249
- Metaphoric communications, 22
- Metes, G., 212, 224
- Methods, 254
- Metrics, 286
- Mexico, 46
- MFG/PRO, 230
- Microsoft, 57, 90
- Microsoft Access, 17, 97
- Microsoft SQL server, 91
- Middleton, P., 240, 264
- Miranda IM, 72, 126, 128
- MIS, 222, 225, 254
- Mislearn, 48
- Mobile computing software project, 142
- Modeling, 178, 226–227, 229, 231–233
- Models, 6, 15, 19, 20, 38, 49, 53, 57, 89, 153, 173, 198, 207, 261, 268
- Moderator, 192
- Moneyless world, 137
- Motivation, 148
- Mozilla, 68
- MRP, *see* Manufacturing resource planning
- MS SQL Server, 17
- Multiple project, 20
- Nammik, 117, 123–126, 128
- Nammik architecture, 126–127
- Natural language, 151
- Nawrocki's experiment, 175
- Nawrocki, J., 24, 35, 142, 163, 175, 195

- Negative values, 272
- Network, 144
- Network News Transfer Protocol (NNTP), 111
- Network programming, 111, 125–126
- Nonexistent customer code, 272
- Nonplagiarized code, 94
- Non-zero-point collaboration, 262
- Nosek, J. T., 141, 163, 175, 195
- Novice programmers, 94, 173, 180

- Object classes, 258
- Object-oriented programming, 156
- Object-oriented software system, 254
- O'Brien, K. J., 108, 130
- Observer, 193
- Ohno, T., 50, 54, 279, 289
- Opdyke, W. F., 254, 264
- Open-ended questions, 161
- Open source, 55–58, 60, 62, 64, 66–68, 70, 72, 74, 76, 78, 80, 82–84, 117, 130
- Open-source developers, 66, 70, 75, 77, 81, 110
- Open-source development, 67, 77–78, 111
- Open-source development project, 72–73
- Open-source maturity model, 57
- Open-source practices, 57
- Open-source products, 57, 65, 108
- Open-source projects, 61, 66–69, 71–73, 75, 111, 126
- Open-source software, 55–57, 65, 67–69, 71–73, 75, 108
- Open-source software development, 34, 55, 57, 62–63, 68, 75, 81–82, 84, 111, 145
- Open-source software project, 56–57, 61, 66, 68, 70
- Operating system, 12–13, 67, 271
- Operational processes, 23
- Optimum productivity, 207
- Organizational cultures, 23
- OSS, *see* Open-source software entries
- OSSD, *see* Open-source software development
- Outsourced programmers, 45, 47, 189

- PacMan, 43
- Paid by bugs, 270
- Pair design, 156–157
- Paired experienced programmers, 135
- Paired programmers, 133, 135, 137, 139, 157, 161, 186
- Paired team, 159, 162
- Pair groups, 153
- Pair jelling time, 207
- Pair learning, 137
- Pair programming, 32, 58–59, 131–148, 152–163, 166–173, 175–182, 184–185, 187, 192, 195–196, 207, 260–261, 282, 292
- Pair programming group, 170
- Pair programming practice, 135, 162
- Pair programming productivity, 162
- Pair programming teams, 80, 133, 161
- Pair-solo rhythm, 184–185
- Pair work, 137
- Panko, R. R., 261, 264
- Parkinson's law, 18
- Parrado, N., 197, 224
- Partner exchange, 158
- Partner rotation, 159
- Paste code, 88
- PAT, *see* Programming aptitude test
- Pattern theory, 100
- Patterns, 100–101
- Paulk, M., 240, 264
- Pavlicek, R. C., 67, 83
- PDCA, 14–16
- PDCA cycle, 14–15
- Peer reviews, 258
- People communications, 238
- People discipline, 288
- People-focused, 194
- People network, 111, 122
- People over process, 199
- Performance tuning, 69
- Perry, J. W., 261, 264
- Personal interest, 68
- Personality test, 43, 45
- Personality traits, 38, 148
- Personal preference, 11
- Personnel turnover, 138
- Petouhoff, N. L., 190, 195
- Petty, D. J., 239, 263
- Plagiarism, 87, 89, 90, 93, 95, 98–100, 102, 107–110, 112, 114, 117, 128–129
- Plagiarism programming, 87–88, 90–100, 102, 104–110, 112, 114–118, 120, 122, 124–130
- Plagiarism programming team, 96
- Plagiarized assignments, 89
- Plagiarized code, 94–95
- Plagiarizing programmers, 91, 93, 97
- Plan to cost, 7
- Planning, 5, 226, 268
- Platforms, client/server, 12
- PL-SQL, 102
- Plug-ins, 126
- Poff, M. A., 180, 187, 196
- Point-of-sales (POS), 12, 151
- Polymorphisms, 156

- POP3, *see* Post Office Protocol Version 3
- Poppendieck, M., 20, 35
- Poppendieck, T., 20, 35
- POS, *see* Point-of-sales
- Positional diversity, 205
- Post Office Protocol Version 3 (POP3), 110
- Post, T. J., 11, 35
- Postmortem review, 4
- PowerBuilder, 12
- Prefactoring, 257
- Premature bad solutions, 51
- Preston, J., 52, 54
- Preventing teaming problems, 211
- Prison experiments, 165
- Probability distribution, 189–190
- Problem solving, 39
- Procedural algorithms, 155–156
- Process(es), 5, 15, 20, 24, 28, 34, 58–61, 63, 68, 82–83, 113, 199–201, 240, 253–254
- Process-centered, 38
- Process-driven, 194
- Process over people, 199
- Process, people, paper, and product, 268
- Product backlogs, 227
- Product integration, 57
- Production phase, 206
- Productive team, 207, 215
- Productivity, 63, 74–75, 134–135, 139–140, 153, 162–163, 169, 172–174, 177, 188, 193–194, 206–207, 209, 278–280
- Products, 4, 7–10, 19, 22, 34, 52–53, 59–61, 63, 65–67, 69, 77–78, 105, 116–117, 183
- Professional service, 57
- Program execution, 270
- Program specifications, 273
- Programmer(s), 46–49, 70–72, 95–97, 103–108, 132–136, 139–142, 156–159, 167–172, 177–181, 185–193, 232–234, 246–247, 259–262, 280–284
- Programmer productivity, 140
- Programming, 25, 40–41, 58, 60–61, 89, 90, 97–100, 103–105, 138–139, 153–154, 165–170, 172–182, 184–188, 192–194, 273
- Programming activities, 233, 288
- Programming aptitude test (PAT), 43
- Programming as execution, 5
- Programming code, 10
- Programming design, 39, 133, 155
- Programming design alternatives, 155
- Programming efforts, 152, 227
- Programming exercises, 174
- Programming experiments, 166, 174
- Programming languages, 12–13, 89, 268, 275
- Programming logic, 232, 256, 270
- Programming modules, 140, 271
- Programming paradigms, 41, 202
- Programming practices, 41, 76, 140
- Programming problems, 40, 101, 150, 172, 175–176
- Programming process, 5
- Programming productivity, 139, 148, 162, 169
- Programming rework, 49
- Programming skills, 41, 45, 47, 97
- Programming solutions, 177, 281
- Programming tasks, 6, 59–61, 95, 134, 142, 148, 152, 158, 173, 181
- Programming techniques, 93, 283
- Progress reports, 23
- Project, 3–8, 14–15, 18–21, 47–49, 51–53, 60–61, 65–68, 70–72, 138, 142–145, 187–191, 198–202, 208–209, 216–223
- Project charter, 198
- Project cycle, 207–209, 226
- Project development team, 201
- Project goals, 194, 220
- Project leader, 12, 29, 70, 200, 213
- Project lifecycle, 212
- Project management, 10, 138, 161, 198, 218, 221
- Project management design, 42
- Project management model, 4
- Project management paradigm, 6
- Project management scheduling, 227
- Project management tools, 218
- Project managers, 4, 6, 7, 18, 46–47, 138, 168, 171, 175, 188–191, 201, 216–217, 221–223, 266–267, 283–284
- Project meeting, 201, 217
- Project members, 218, 221
- Project mismanagement, 3
- Project participants, 71
- Project plan, 5–7, 15, 20, 23, 77, 142, 189, 206, 209, 220, 227, 230, 284
- Project planning, 8, 188, 231, 284
- Project requirements, 6
- Project resources, 5, 218, 266–267
- Project schedule, 18, 77, 201, 220–221, 267
- Project skill demand curve, 208
- Project structure, 216
- Project team, 69, 83, 112, 188, 190–191, 200–203, 205–207, 221–222, 238, 285
- Project team composition, 205

- Project team members, 201, 223
Project time, 188, 207
Project tracking, 77, 81, 284
Prosumer, 70
Prototype, 4, 14–16, 23, 65, 69, 73, 237
Psychology of programming, 38–39, 175
Pugh, K., 257, 264
Pygmalion in the classroom, 37
Python code, 276
- Quality assurance (QA), 272
Quality checking (QC), 272
Quality improvement, 278
Quasi-experiment, 176
Quick solutions, 88, 262, 269, 273,
275–276, 284
- Raccoon, L. B. S., 20, 35
Radiotherapy, 4
Rapid iterative development cycle, 284
Rapid releases, 63, 66, 72, 74–75, 77, 81
Rapid software process improvement, 282–
283, 285, 287
Rapidly changing requirements, 251
Rause, V., 197, 224
Raymond, E. S., 65, 83
Readability, 10, 25, 92, 156, 254, 256, 258–262
Redesign, 105
Redistributing classes, 254
Redundancy, 236
Refactoring, 25, 69, 79, 89, 92, 105–107, 179, 234,
254, 256–263, 274–275, 277, 280, 286
Refactoring techniques, 258
Regression testbed, 277
Regression testing, 23
Reicher, S., 165, 195
Release patches, 16
Releases, 16, 23, 29, 67, 69, 70, 72–75,
77, 270
Remote access, 102
Remote desktop, 193
Repeat programming, 172–173
Repeat-test, 95
Repetitious programming tasks, 293
Repetitive products, 246–247
Report writing, 152
Requirement(s), 5, 10, 14, 16–17, 24, 29, 147, 151,
228, 234, 246–255, 267–269, 271, 273
Requirement specifications, 60
Requirements complexity, 248–249, 251, 253
Requirements engineering, 151
Requirements management processes, 22
Requirements management, 8, 71, 152
Rescue practice, 216–217
Rescuing software project, 218
Resource(s), 5–7, 15, 58, 60, 83, 144, 152, 198,
208, 219–220, 222, 227, 230, 281
Resource allocations, 151
Return on investment, 22
Reusability, 235, 254, 278
Rework, 235
Rhythm(s), 24–28, 30–33, 117, 125–129, 159–
160, 178–179, 181–188, 194,
202–204, 206–207, 241, 274–275,
281–283, 291–293
Rhythm for challenging tasks, 293
Rhythm for late projects, 192
Rhythm of the groups, 162
Rhythm of software practices, 26
Rhythm of triple programming, 194
Rhythmic pair programming, 158, 160
Rhythmic problem management cycle, 218
Right rhythms, 24, 34, 292
Right timing, 204
Rigorous, 194
Rigorous development processes, 5
Ringelmann effect, 149
Risk, 7, 11, 24, 29, 35, 38, 103, 136–138, 177, 199,
203, 267
Risk control, 218
Risk factors, 6, 15, 189
Risk management, 218, 223
Robert, M., 145, 163
Role(s), 10, 18–19, 57, 70–72, 82, 131–132, 135,
158, 168, 192, 198, 221, 226, 237
Role exchange, 158
Roles of programmers, 70, 82
Root cause, 51
Root cause analysis, 215
Rorschach inkblot, 41
Rosenberg, D., 169, 196, 289
Rosenthal, R., 37, 54
Rough-cut design, 106
Royce, Walker, 9, 35
Royce, Winston, 5, 35
Runaway software project, 218
Running project team, 201
- Saiedian, H., 283, 288
Sales-and-distribution system (SDS), 118–120
Sashimi model, 20
Schedule, 15, 141
Schedule-based, 194
Scheduling slippage, 205

- Schoemaker, P. J. H., 262, 264, 273, 289
- Schonberger, R., 239, 264
- Schwaber, K., 161, 163, 203, 224
- SCM, *see* Supply chain management
- Scrum, 14, 203, 227, 288
- Scrummage, 203
- Scrummage meeting, 14
- SDS, *see* Sales-and-distribution system
- Self-fulfilling prophecy, 37
- Self-organizing team, 79, 161, 202–204, 210, 214–215
- Semantic algorithm analysis, 154
- Server managers, 102
- Shalloway, A., 105, 130
- Shared-intelligence collaboration, 87
- Shell, M. M., 40, 54
- Shewhart's closed-loop model, 14
- Short descriptive requirements, 22
- Side-by-side programming, 132, 166–167, 282
- Sign cards, 159
- Signed confirmations, 4
- Simple design, 231, 275
- Simple Mail Transfer Protocol (SMTP), 110
- Single pair programming, 158–159, 162, 166, 179–180
- Single-phase analysis, 6
- Sjoberg, D., 51, 54
- Skilled programmers, 45, 105, 107
- Slack, 144
- Slack-time, 20
- Slow-paced, 194
- Small-lot-size user requirements, 246
- Smart software team, 279
- SMTP, *see* Simple Mail Transfer Protocol
- SNA, *see* Social network analysis
- Snow, A. P. 219, 224
- Social network analysis (SNA), 91
- Software, 3, 4, 8–10, 14, 22–24, 29–31, 33–34, 48–50, 54–55, 60–61, 65–67, 69, 73, 266–268, 270–271
- Software activities, 41
- Software applications, 13, 66, 137, 232, 246, 266
- Software architects, 226
- Software capability, 199
- Software cloning, 66–68
- Software complexity, 183, 223, 237, 250, 266
- Software configuration, 88, 112, 286–287
- Software configuration management, 77, 112, 286–287
- Software construction, 130, 233, 238
- Software design, 43, 226–8, 231–232, 249, 258, 268
- Software design methodology, 248
- Software design teams, 224
- Software developers, 3, 10, 16, 24, 33, 49
- Software development, 4, 9, 33–35, 38–39, 57, 60–61, 82–83, 185–186, 188, 199, 200, 209, 237–238, 240–241, 266
- Software development activities, 5, 268
- Software development methodologies, 22, 77
- Software development processes, 4, 10, 56
- Software development projects, 3, 57, 265, 267
- Software development rhythms, 3, 24–25, 27, 29–31, 33–34, 37, 52–53, 55, 82, 87, 114, 129, 194, 291
- Software development tasks, 152
- Software development teams, 212
- Software disciplines, 129
- Software engineering, 3, 35, 54, 130, 288
- Software functionalities, 22, 232, 271
- Software house, 12, 21, 73, 237, 270
- Software inspections, 138–139
- Software installation, 97
- Software integration, 24, 152
- Software leader, 28
- Software maintenance, 60, 156, 166
- Software management, 10, 52
- Software managers, 6, 9, 43, 45, 60, 82, 200, 215, 283
- Software methodologies, 38, 47, 61, 194, 283
- Software module, 75, 152
- Software outsourcing, 45
- Software paradigms, 29, 34, 78, 283
- Software practices, 6, 24–26, 32, 38, 41, 43, 47, 49, 52, 54–55, 58–59, 74–77, 79–82, 283
- Software principles, 175, 194
- Software problems, 10
- Software process, 34, 58–60, 199, 241, 263
- Software process improvement, 58, 240, 289
- Software products, 7, 40, 52, 68, 70, 72, 129, 174, 198, 248
- Software professionals, 25
- Software program managers, 52
- Software project, 6–8, 10–12, 48–49, 51, 61, 71–73, 138, 142, 175, 199–202, 207, 212–213, 215–218, 248–249
- Software project community, 76
- Software project dynamics, 195
- Software project experience, 34
- Software project failures, 71, 175, 222, 253
- Software project management, 14, 48, 60–61, 76, 145
- Software project manager, 11
- Software project participant roles, 72

- Software project planning, 82
- Software project status, 224
- Software project teams, 198
- Software quality, 49, 65, 67–68, 103, 136, 138, 140–141, 174, 176–177, 258, 260, 266–267, 271, 286
- Software requirements, 5, 64, 146, 202, 230, 251, 264, 267
- Software rhythms, 34
- Software solutions, 47, 63
- Software teams, 6, 7, 10–12, 15, 17–19, 22–25, 28–29, 31, 33–34, 38–39, 41, 46–48, 74–75, 81–83, 152–153
- Software team's productivity, 209
- Software testing, 18, 23
- Software tools, 39, 207
- Software verification, 271
- Solo groups, 153
- Solo programming, 33, 133, 138–139, 142, 144, 161, 166–167, 169–170, 177–178, 180–182, 184–187, 194, 259
- Soloway, E., 94, 130
- Solution(s), 6, 63–64, 104, 114, 145–146, 150, 168–169, 173, 176–177, 262, 269, 273, 275–276, 281–282
- Solution-oriented, 102
- Sommerville, I., 58, 83, 108, 130
- Sonnentag, S., 198, 224
- Source code, 65, 67–68, 73–76, 78, 88, 108, 112, 210, 233, 259
- Spiral model, 15–16, 35
- Sprint, 202
- SQL database programming, 93
- SQL, *see* Structured query language
- Staff, 17, 109, 116, 131, 187, 192, 205–206, 223, 276, 292
- Staff turnover, 107
- Stalaker, A. W., 39, 54
- Stamelos, I. S., 67, 83
- Standardized process, 241
- Standish Group, 253
- Standup meeting, 24–25, 27, 78, 150, 160–161
- Stanford prison experiment, 165
- Stasser, G., 136, 163
- Static, 194
- Statistical analysis, 258
- Statistical process control, 14
- Stave chart, 25
- Steiner, I. D., 152, 163
- Stephens, M., 169, 196, 289
- Stock, make to, 244
- Story cards, 159, 227–233, 246, 266, 273, 275–276, 284
- Strategic planning, 132
- Strong rhythms, 241, 281, 283
- Structure of a rhythm, 28
- Structure phase, 206
- Structured query language (SQL), 10
- Stubblefield, W., 155, 163
- Student programmers, 43, 126
- Style consistency, 259
- Subcomponents, 7, 8
- Submodules, 50, 69, 74, 95, 105, 110, 141–142, 152, 177, 273
- Subprograms, 87–88, 149
- Substantial programming reworks, 49, 50
- Substantial redesign, 232
- Subtasks, 60, 65, 132, 148–149, 151, 181–184, 192
- Subteams, 19, 113, 204, 212–213
- Suckers, 210
- Supply chain applications, 40
- Supply chain management (SCM), 110, 216
- Support pair programming productivity, 166
- Sustaining pair programming, 179–180
- Sutton, J., 240, 264
- Swire group, 117
- System analysts, 10, 19
- System design, 41
- System integration, 5, 152, 205, 230, 250
- System modules, 251
- System requirements, 147
- Tacit knowledge, 31
- Tailored processes, 240–241
- Talented programmers, 40, 140
- Task(s), 5, 7, 8, 18, 20–21, 92–93, 133–134, 136, 140–144, 148–153, 191–192, 199, 202–203, 209–211, 227
- Task execution, 238
- Task switching, 20
- Taylor, D., 216, 224
- TCP/IP, *see* Transmission Control Protocol/Internet Protocol (TCP/IP)
- TDD, *see* Test-driven development
- Teaching, 168
- Team, 23–25, 28–34, 38–43, 45–49, 63–65, 68–70, 111–116, 150–153, 186, 190–192, 198–209, 211–213, 237–238, 285–287
- Team collaboration, 198, 238, 283
- Team communications, 59, 92, 281
- Team coupling, 79, 80
- Team cultures, 38, 42, 48, 92, 166, 201
- Team development, 201
- Teaming, 69, 167, 198–200, 202, 212, 218–219

- Teaming principles, 215
- Teaming problems, 207
- Teaming relations, 21
- Team-in-team approach, 204
- Team leaders, 39, 47, 64, 185–186, 217, 222, 281
- Team location, 77, 79
- Teammates, 139, 201, 210, 233
- Team members, 6, 19, 24, 27, 41, 64–65, 76–79, 138, 151, 190–192, 201–203, 205, 209–210, 281–282
- Team of programmers, 156, 247
- Team organization, 60, 198, 203
- Team pair programming, 158–160, 162, 166, 204
- Team pair programming productivity, 159
- Team performance, 148, 151–152, 205, 209, 286
- Team productivity, 135–136, 153, 208–210
- Team programming, 75, 134, 162
- Team progress, 185, 228
- Team size, 77, 79, 214
- Team software development, 76
- Team software process (TSP), 198
- Team structure, 205, 212–214, 222–223
- Team velocity, 228–230
- Teamwork, 4, 40, 46, 167, 210
- Technical programming, 205, 288
- Technical staff, 12
- TechTrans, 11
- Telelogic's Logiscope, 67
- Test, 39, 40, 43, 90–91, 96–97, 99, 100, 108, 112–114, 116, 125–126, 174, 179, 262, 266–269, 271–281
- Test case, 116, 140, 174, 256, 258, 262, 266–273, 275–276, 279, 281, 293
- Test case requirements, 271
- Test code, 268
- Test-driven development (TDD), 104, 185, 267–268, 274–275, 278, 280–281, 283, 284, 286–288, 292
- Test environment, 97
- Tester, 10
- Test-first programming, 79, 267, 269–271, 273–274, 279, 289
- Test-first refactoring, 262
- Test-first thinking, 278–279, 282, 286
- Testing, 268
- Test-last programming, 270, 279
- Thematic rhythms, 25
- Theme, 291
- Therac-25, 4
- They'll love it, 235
- Thompson, Ed, 22, 110, 198
- Thompson, K., 67, 83
- Thread programming, 283
- Time box, 18
- Time-critical tasks, 161
- Timeframes, 12
- Time-to-market, 177
- Time-wasting, 203
- Tindale, R. S., 152, 163
- Toffler, Alvin, 70
- Tools, open-source, 277
- Torment your customers, 235
- Total cost concept, 231
- Tourwe, T. A., 234, 262, 264
- Toyota Motor Company, 239
- Toyota production system (TPS), 239
- Traditional project management, 60, 227, 284
- Traditional project planning, 227–228
- Traditional software development, 199, 229, 246, 287
- Traditional team structure, 203–204
- Traffic light reporting, 219
- Training, 57, 218
- Training manuals, 60, 71
- Transact-SQL, 96
- Transmission Control Protocol/Internet Protocol (TCP/IP), 91
- Transparency, 211
- Triple programming, 132–133, 145–146, 162, 168–169, 177–178, 192–194
- Triplet programming, 162
- TripLog, 17
- Trot, J., 105, 130
- Troubled late IT projects, 218
- Troubled-late projects, 190–191
- Truck number, 138, 191
- TSP, *see* Team software process
- Tuckman model, 206
- Tversky, A., 11, 35
- UAT, *see* User acceptance test
- Ugly customers, 22
- Ugrammers, 70–71
- UML, *see* Universal modeling language
- Unit costs, 7
- Unit test, 5, 114, 184, 258, 271–279, 281–282, 284–287
- Unit test cases, 273, 276–278, 282, 286–288
- Unitarily conjunctive, 152
- Universal modeling language, 10, 233
- Unix, 67, 84, 198
- Unpredictable programming changes, 133

- Update products, 183
- Upfront design, 237–239, 247, 255
- USENET, 138
- User(s), 8, 14, 17, 21, 65, 70–73, 147–148, 174–175, 202–203, 221, 225–226, 232–235, 250–251, 253–255
- User acceptance tests, 5, 22, 212–213, 271
- User interfaces, 74, 89, 91, 95, 226, 233, 268
- User requirement log, 274
- User requirements, 6, 11, 41, 58, 93, 108, 112–113, 147, 151, 226–227, 237–238, 250–251, 253–254, 268–269
- User roles, 71–72
- UserSession, 154
- User specifications, 233
- User stories, 8, 22, 202, 274
- User test cases, 266

- Value-added tax (VAT), 105
- Variables, 254
- VB, *see* Visual BASIC
- VB.net, 89
- VCD, *see* Video compact disk
- Velocity, 228–230, 284
- Video compact disk (VCD), 116
- Virtual team, 212
- Virtuosi, 40
- Visual BASIC (VB), 89, 90
- Visual software design, 232–233

- Wald, R. M., 156, 163
- Walter, B., 24, 35
- Warehouse application, 174
- Waterfall management, 7
- Waterfall model, 5, 6, 10, 12, 16, 28–29, 41, 227, 243, 288

- Web-based content management system, 117
- Web-based CRM, 187
- Web design, 97
- Welch, J., 200, 224
- Welch, S., 200, 224
- Whitebox testing, 108
- Wiegiers' and Blackburn's data, 249
- Wiegiers, K. E., 248–249, 264
- Williams, L. A., 133, 138, 139, 141, 162, 163, 175, 196, 207, 211, 224
- Windows 95, 91
- Windows 98, 271
- Windows programming, 90
- Windows Server 2003, 57
- Within-classes, 256–257
- Wojciechowski, A., 24, 35, 142, 163, 175, 195
- Womack, J. P., 239, 264
- Work products, 73, 183–185, 240, 247, 286
- Working, 168
- Working software, 12, 23, 60, 68, 79, 106
- Work-in-progress, 243, 244

- XP, 25, 34, 80–81, 132, 168, 196, 283, 288–289
- XUnit, 277

- Yeh, Y. J., 198, 205, 224
- Yuhas, C. M., 21, 35

- Zahran, S., 283, 289
- Zero-defect software, 271
- Zero-point collaboration, 64
- Zimbardo, P. G., 165, 196

