

---

# Index

## A

Ada, 87–88  
AD/Cycle, 26  
Ad hoc, 163  
Advocacy research, 148–150  
Agile development, 13, 83–84, 86, 111, 162, 163  
Algorithmic approaches, 28–31  
Analysis paralysis, 72  
Applied Computer Research (ACR), 26–27, 99–100  
Assembly language, 87–89, 139–143  
Audience, 3, 182

## B

Basili, Vic, 60, 71–72, 107, 109, 111–112, 156  
“Bathtub” shape (to software maintenance cost curve), 177–180  
Beck, Kent, 19, 38, 141  
Benson, Miles, 22  
Biased errors, 134–135, 172  
Biggerstaff, Ted, 47, 50–51  
Boddie, John, 30  
Boehm, Barry, 12–13, 15, 71–72, 107, 116, 132, 136–137, 139  
Bollinger, Terry, 13  
Bosch, Jan, 49  
Bowen, Thomas B., 128–129  
Branch testing. *See* Logic paths

Brooks, Fred (and Brooks’s Law), 16–17, 20–22, 55, 60, 103, 111–112  
Brossler, P., 111–112  
Bucher, 13–14  
Bush, Marilyn, 107

## C

Capability Maturity Model (CMM), 11–12, 138  
CASE tools, 20, 25–27, 34–35, 101  
Chapin, Ned, 122  
Checkout. *See* Error–removal  
Churchill, Winston, 89  
Cleanroom (testing approach), 173–174  
Clerical (software work as), 60–63  
Cluster (of errors), 135–136, 172  
COBOL, 87–89  
Cockburn, Alistair, 163  
Coding, 65, 84–89, 139–143  
Cognitive processes, 82  
Cole, Andy, 30, 40–42, 70  
Collofello, Jim, 109  
Colter, Mel, 40–42  
Communism, 161  
Complexity, 10, 58–63, 77–79, 81–84, 174, 185  
Component. *See* Reuse  
Comprehension. *See* Understanding  
Computer Sciences Corp, 186  
Corbi, T.A., 183

Creative (software work as), 60–63, 66  
 Crunch mode, 27, 30  
 Curmudgeon, 6  
 Curtis, Bill, 81–84

**D**

Dangerfield, Rodney, 98  
 Davis, Al, 9, 11, 13, 15, 22, 72, 136–137  
 Death march, 27, 30  
 Debug code, 103–104  
 Debuggers, 19, 97–100, 175  
 DeGrace, Peter, 165–166  
 Deimel, Lionel, 183  
 Dekleva, Sasa, 125–126  
 DeMarco, Tom, 12–13, 17–19, 154, 158, 159, 165–166  
 Denver International Airport, 36, 69, 71  
 Department of Defense (DoD), U.S., 12, 128–129  
 Derived requirements, 76–79  
 Design, 65–67, 76–84, 139–141  
 Design envelope, 52  
 Design pattern. *See* Pattern  
 Disconnect (between programmers and managers), 34, 39–42, 185  
 Diversity (differences among projects and domains), 46–47, 162–163  
 Domain dependence, 48–49  
 Domain-specific languages, 87–89  
 Dyer, Mike, 79

**E**

Early adapters, 21  
 Easy-part first (design), 83  
 Ebner, Gerald, 79  
 Education (of software engineers and computer scientists), 181–184  
 Efficiency, 130–132, 139–145  
 Egoless programming, 114, 160–161  
 Endres, A., 136–137  
 Enhancement, 117–120, 177–180  
 Enterprise resource planning system, 24, 53  
 Error-free (software product), 67, 74–76, 93–95, 108, 137–139  
 Error removal, 65–67, 74, 99–115, 134–139  
 Errors of combinatorics, 96

Errors of omission, 96  
 Estimation, 10, 27–43, 58, 132–134, 167–169, 185  
 Evaluative research, 20, 148–150  
 Exception case handling, 172  
 Experience factory, 111–112  
 Experience level (of software field), 111–112  
 Extreme Programming, 19, 20, 23–24, 38, 52, 70, 72, 80, 83–84, 86, 111, 141  
 Eyeballs, 174, 177

**F**

Failure (of a software project), 39–42, 68  
 Fallacies, 151–184  
 Families (of related systems). *See* Domain dependence  
 Fault-tolerance, fault-tolerant programming, 108, 135  
 F-Book, 4–5  
 Feasibility, 42–43, 82  
 Feature point, 29  
 Fenton, Norman, 149–150  
 Fit and finish, 131  
 Fjelstad, Robert K., 121–124  
 Formal specification, 68–72  
 Formal verification, 108–109, 173  
 Fortran, 87  
 Fourth generation languages, 20  
 Fowler, Martin, 119, 141  
 Front-end loading, 90  
 Function point, 29  
 Fuzz papers, 175–177

**G**

Gamma, Erich, 56–57  
 Gang of Four, 57  
 Gates, Bill, 124, 184  
 Genetic testing (dynamic generation of test cases), 173–174  
 Glass, Robert L., 11, 16, 20, 22, 27, 45, 63, 71, 76, 79, 84, 87, 90–91, 93–95, 96–97, 104, 107, 109, 111–112, 120, 129, 131–132, 135, 137, 138, 148–150, 159, 162–164, 166, 176, 180, 183  
 GQM (Goals/Questions/Metrics), 156  
 Grady, Robert B., 20–22  
 Gramms, Timm, 134–135

**H**

Halstead, Murray, 156, 158  
 “Hard drives out soft”, 18  
 Hard-part first (design), 81–84  
 Hardware envy, 21  
 Hardy, Colin, 164, 167  
 Hetzel, Bill, 156–158  
 Heuristic, 58, 82  
 Highsmith, James A., 13–14, 163–164  
 HOL (High-Order Language), 19, 141–143  
 Human engineering, 130–132  
 Hunt, Andrew, 88–89  
 Hype, 19–22, 44, 137, 148, 149, 176, 185

**I**

IEEE, 47  
 Individual Differences, 14–16  
 Inspection, 59, 67, 104–115, 173, 174–177  
 Intellectual (software work as), 60–63, 66

**J**

Jacobson, Ivar, 62–63  
 JAD (Joint Application Development), 69  
 JARGON, 51  
 Java, 144  
 Jazayeri, Mehdi, 49  
 Jeffery, D.R., 41–42  
 Jenkin, Steve, 111  
 Jones, Capers, 31, 162–164, 168–169

**K**

Kaner, Cem, 103, 139  
 Kerth, Norman, 110–113  
 Kitchenham, Barbara, 30–31

**L**

Laggards, 21  
 Lammers, Susan, 124, 184  
 Landsbaum, Jerome B., 41–42, 117  
 Learning curve, 23–26  
 Lederer, Albert, 34–35  
 Lessons learned, 24  
 Libraries (reuse-in-the-small), 43–45  
 Lientz, Bennet, 117  
 Life cycle, software, 65–126, 171–180

Linberg, K.R., 39–42  
 Lines of code (LOC), 29–30, 40, 106, 167–169  
 Lister, Tim, 12–13, 17–19  
 Little Moron (joke), 12  
 Logic paths, 93–97  
 Loyal Opposition (in *IEEE Software*), 6

**M**

Mainframe, 43  
 Maintenance, 52–53, 59, 65, 115–126, 177–180, 183  
 Malpractice, 30  
 Management, 9–63, 155–169  
 Management by schedule, 37–38  
 Management by schedule (alternatives to), 37  
 Maryland, University of, 186  
 McBreen, Pete, 15, 48–49, 86–87, 89, 163  
 MCC (Microelectronics and Computing Consortium), 81  
 McCall, J., 132  
 McClure, Carma, 47  
 McConnell, Steve, 17  
 McGarry, Frank, 54–55  
 Measure, Metric, 155–158  
 Methodology, 11, 164–167  
 Michael, Christopher C., 173–174  
 Microsoft, 36, 75–76  
 Miller, Barton P., 175–176  
 Mills, Harlan, 154, 173–174, 184  
 Minimum standard toolset, 26–27  
 Modeling, 71, 72  
 Modifiability, 127, 130–132  
 Mohanty, S.N., 28, 31  
 Myers, Glenford, 15–16, 109  
 Mythical Man–Month, 17

**N**

N-version programming, 135  
 NASA (especially NASA-Goddard), 35–36, 46, 48, 54–55, 75–76, 155, 186  
 Not invented here (NIH), 26, 46

**O**

Object-orientation, 20  
 Open source, 53–54, 99–100, 165, 174–177  
 Operating systems, 19

Opportunistic (design), 81–84  
 Optimal design, 81–84  
 Optimization, 143–145  
 Oracle (correct answers for testing), 172  
 Outsourcing, 85

**P**

Pair Programming, 19  
 Parnas, David, 60, 83–84  
 Pattern, 55–58  
 People, 10, 11–19, 33–35, 58, 160–161  
 Peopleware, 12–13, 17–19  
 Persistent software errors, 74–76, 96–97  
 Pirsig, Robert M., 127, 129  
 Plauger, P.J., 163–164  
 PL/1, 88  
 Portability, 128, 120–132  
 Post-delivery reviews, 110–113  
 Potts, Colin, 149–150  
 Power of Peonage, 10, 11  
 Practical Programmer (in *Communications of the ACM*), 6  
 Prentiss, Nelson H., 143  
 Pressman, Roger, 32  
 Preventive maintenance, 118–119, 141  
 Primitives (for coding), 84–87  
 Principles (of software engineering), 9, 11  
 Priority (of software errors), 76  
 Procaccino, J. Drew, 41  
 Process, 11–12  
 Productivity, 17–19, 23–25  
 Prototyping, 69, 71

**Q**

Quality (of the software product), 17–19, 37, 137–145, 158–159

**R**

Radice, Ronald A., 108–109  
 Ramping up, 86  
 Random testing approaches. *See* Statistics-driven testing  
 Rational Software, 62  
 Reading-before-writing, 181–184  
 Reality, 187

Refactoring. *See* Preventive maintenance  
 Regression testing, 172  
 Reifer, Donald J., 47–48  
 Reilly and Maloney, 147  
 Reliability, 128, 129–132, 133–134, 134–139  
 Requirements, 31, 65–67, 67–76, 132–134  
 Requirements-driven testing, 92–93  
 Requirements explosion, 58, 92  
 Requirements traceability, 77–79  
 Research (software engineering), 147–150  
 Retrospectives. *See* Post-delivery reviews  
 Reuse, 10, 43–58  
 Reviews, 67, 72, 76, 104–115  
 Rich, Charles, 101–103  
 Rifkin, Stan, 106–107  
 Rigor vs. relevance (in research), 61  
 Risk-driven testing, 92–93  
 Rolling totals, 82, 85  
 Rombach, Dieter, 155, 158  
 RPG, 87  
 Rubey, Raymond, 13–14, 142–143, 144–145  
 Rules of three. *See* Reuse  
 Runaway projects, 27–43, 67–71

**S**

Sackman, H., 14–16  
 Sanden, Bo, 163–164  
 SAP. *See* Enterprise resource planning system  
 Satisficing, 83–84  
 Schwartz, Jules, 15–16  
 Seattle University, 78, 159  
 Severity (of software errors), 74–76, 137–139, 175  
 Share, 44–45  
 Shelfware, 25, 98, 101  
 Silver bullet, 20–22, 137  
 Simon, Herbert, 83–84  
 Simulation, 71, 82  
 Smidts, Carol, 138–139  
 Software crisis, 68  
 Software Engineering Institute (SEI), 11–12, 186  
 Software Engineering Laboratory (SEL). *See* NASA  
 Software Practitioner, 88–89  
 Software science, 156  
 Soloway, Elliott, 81–84  
 Spiral (life cycle), 66, 91

- SQL, 87  
 Statistics-driven testing, 92–93, 171–174  
 Structure-driven testing, 92–97  
 Structured methods, 149  
 Success (of a software project), 39–42  
 Sullivan, Daniel J., 177, 180  
 Sweeney, Mary, 103  
 SYMPL, 87  
 Systems analyst, 73–74, 85  
 Systems engineer, 73–74, 85
- T**
- Taylor, Dennis, 60  
 Teams, 160–161  
 Techniques, 10–12, 19–22, 23–25, 161–167  
 Testability, 130–132  
 Test automation, 98–103  
 Test coverage, 59, 91–104, 171  
 Test coverage analyzer, 23–25, 91–104  
 Testing, 65–67, 72, 91–104, 171–174  
 Test managers, 172  
 Test tools, 97–100  
 Theory, 187  
 Thinking traps, 134–135  
 Thomas, William, 54–55  
 Through a Glass, Darkly (in ACM’s SIGMIS database), 6  
 Tichy, Walter F., 149–150  
 Tools, 10–12, 19–22, 23–25, 25–27, 161–167  
 Tracz, Will, 47–48, 51
- U**
- Understanding, Understandability, 120–124, 130–132, 181–184  
 Unified Modeling Language (UML), 62
- Unstable requirements, 28  
 Used program salesman. *See* Tracz  
 User satisfaction, 132–134
- V**
- Van Genuchten, Michiel, 30, 70  
 Verification and validation, 90  
 Vessey, Iris, 149–150, 163–164  
 Visser, Willemien, 56–58  
 Vlasbom, Gerjan, 164, 167
- W**
- Waterfall (life cycle), 65–67  
 Web software, 86  
 Weinberg, Gerald, 22, 25, 43, 136–137, 154, 161  
 Wieggers, Karl, 27, 76, 83–84, 107, 112, 115, 123–124, 164–167  
 Williams, Laurie, 19  
 Wisdom, 111–112  
 Woodfield, Scott, 59–60  
 Workspace (for programmers), 17–19  
 Writing-before-reading, 181–184
- Y**
- Y2K (year 2000) problem, 116  
 Yale University, 81  
 Yourdon, Ed, 30, 163–164
- Z**
- Zealots, 24, 175, 177  
*Zen and the Art of Motorcycle Maintenance*, 127, 129  
 Zhao, Luyin, 99–100, 175–177

