

CONTENTS

List of Figures	xvii
List of Tables	xxi
Preface	xxv

PART I OVERVIEW AND BASICS

1 Overview	1
1.1 Meeting People's Quality Expectations	1
1.2 Book Organization and Chapter Overview	4
1.3 Dependency and Suggested Usage	7
1.4 Reader Preparation and Background Knowledge	9
Problems	11
2 What Is Software Quality?	13
2.1 Quality: Perspectives and Expectations	13
2.2 Quality Frameworks and ISO-9126	16
2.3 Quality, Correctness, and Defects	18
2.3.1 Definitions: Error, fault, failure, and defect	18
2.3.2 Concepts and relations illustrated	19
2.3.3 Correctness-centered properties and measurements	20
2.4 A Historical Perspective of Quality	22
	vii

2.5 So, What Is Software Quality?	24
Problems	24
3 Quality Assurance	25
3.1 Classification: QA as Dealing with Defects	25
3.2 Defect Prevention	29
3.2.1 Education and training	29
3.2.2 Formal method	30
3.2.3 Other defect prevention techniques	31
3.3 Defect Reduction	32
3.3.1 Inspection: Direct fault detection and removal	32
3.3.2 Testing: Failure observation and fault removal	33
3.3.3 Other techniques and risk identification	34
3.4 Defect Containment	35
3.4.1 Software fault tolerance	35
3.4.2 Safety assurance and failure containment	36
3.5 Concluding Remarks	36
Problems	37
4 Quality Assurance in Context	39
4.1 Handling Discovered Defect During QA Activities	39
4.2 QA Activities in Software Processes	41
4.3 Verification and Validation Perspectives	44
4.4 Reconciling the Two Views	47
4.5 Concluding Remarks	49
Problems	50
5 Quality Engineering	51
5.1 Quality Engineering: Activities and Process	51
5.2 Quality Planning: Goal Setting and Strategy Formation	54
5.3 Quality Assessment and Improvement	57
5.4 Quality Engineering in Software Processes	57
5.4.1 Activity distribution and integration	58
5.4.2 Effort profile	59
5.5 Concluding Remarks	61
Problems	62
PART II SOFTWARE TESTING	
6 Testing: Concepts, Issues, and Techniques	65
6.1 Purposes, Activities, Processes, and Context	65
6.2 Questions about Testing	69

6.3	Functional vs Structural Testing: What to Test?	72
6.4	Coverage-Based vs Usage-Based Testing: When to Stop Testing?	76
6.5	Concluding Remarks	81
	Problems	82
7	Test Activities, Management, and Automation	83
7.1	Test Planning and Preparation	83
7.1.1	Test planning: Goals, strategies and techniques	83
7.1.2	Testing models and test cases	84
7.1.3	Preparation of individual test cases	85
7.1.4	Test suite preparation and management	86
7.1.5	Preparation of test procedure	87
7.2	Test Execution and Measurement	88
7.2.1	Overall activities and management	88
7.2.2	Result checking: The oracle problem	89
7.2.3	Test measurement	90
7.3	Analysis and Followup	91
7.4	Activities, People, and Management	93
7.5	Test Automation	95
7.6	Concluding Remarks	98
	Problems	99
8	Coverage and Usage Testing Based on Checklists and Partitions	101
8.1	Checklist-Based Testing and Its Limitations	101
8.2	Testing for Partition Coverage	105
8.2.1	Some motivational examples	105
8.2.2	Partition: Concepts and definitions	106
8.2.3	Testing decisions and predicates for partition coverage	107
8.3	Usage-Based Statistical Testing with Musa's Operational Profiles (OPs)	109
8.3.1	The cases for usage-based statistical testing	109
8.3.2	Musa OP: Basic ideas	110
8.3.3	Using OPs for statistical testing and other purposes	112
8.4	Constructing Operational Profiles	113
8.4.1	Generic methods and participants	114
8.4.2	OP development procedure: Musa-1	116
8.4.3	OP development procedure: Musa-2	118
8.5	Case Study: OP for the Cartridge Support Software	119
8.5.1	Background and participants	120
8.5.2	OP development in five steps	120
8.5.3	Metrics collection, result validation, and lessons learned	122
8.6	Concluding Remarks	123

Problems	124
9 Input Domain Partitioning and Boundary Testing	127
9.1 Input Domain Partitioning and Testing	128
9.1.1 Basic concepts, definitions, and terminology	128
9.1.2 Input domain testing	130
9.1.3 Partition and boundary problems	130
9.2 Simple Domain Analysis and the Extreme Point Combination (EPC) Strategy	132
9.3 Testing Strategies Based on Boundary Analysis	135
9.3.1 Weak $N \times 1$ strategy	135
9.3.2 Weak 1×1 strategy	139
9.4 Other Boundary Test Strategies and Applications	140
9.4.1 Strong and approximate strategies	140
9.4.2 Other types of boundaries and extensions	141
9.4.3 Queuing testing as boundary testing	142
9.5 Concluding Remarks	144
Problems	145
10 Coverage and Usage Testing Based on FSMs and Markov Chains	147
10.1 Finite-State Machines (FSMs) and Testing	148
10.1.1 Overcoming limitations of simple processing models	148
10.1.2 FSMs: Basic concepts and examples	149
10.1.3 Representations of FSMs	151
10.2 FSM Testing: State and Transition Coverage	153
10.2.1 Some typical problems with systems modeled by FSMs	153
10.2.2 Model construction and checking for missing or extra states or transitions	154
10.2.3 Testing for correct states and transitions	155
10.2.4 Applications and limitations	157
10.3 Case Study: FSM-Based Testing of Web-Based Applications	157
10.3.1 Characteristics of web-based applications	157
10.3.2 What to test: Characteristics of web problems	158
10.3.3 FSMs for web testing	159
10.4 Markov Chains and Unified Markov Models (UMMs) for Testing	161
10.4.1 Markov chains and operational profiles	161
10.4.2 From individual Markov chains to unified Markov models (UMMs)	162
10.4.3 UMM construction	164
10.5 Using UMMs for Usage-based Statistical Testing	164
10.5.1 Testing based on usage frequencies in UMMs	164
10.5.2 Testing based on other criteria and UMM hierarchies	165
10.5.3 Implementation, application, and other issues	166
10.6 Case Study Continued: Testing Based on Web Usages	167

10.6.1	Usage-based web testing: Motivations and basic approach	167
10.6.2	Constructing UMMs for statistical web testing	168
10.6.3	Statistical web testing: Details and examples	169
10.7	Concluding Remarks	171
	Problems	172
11	Control Flow, Data Dependency, and Interaction Testing	175
11.1	Basic Control Flow Testing (CFT)	176
11.1.1	General concepts	176
11.1.2	Model construction	178
11.1.3	Path selection	180
11.1.4	Path sensitization and other activities	181
11.2	Loop Testing, CFT Usage, and Other Issues	182
11.2.1	Different types of loops and corresponding CFGs	182
11.2.2	Loop testing: Difficulties and a heuristic strategy	184
11.2.3	CFT Usage and Other Issues	186
11.3	Data Dependency and Data Flow Testing (DFT)	186
11.3.1	Basic concepts: Operations on data and data dependencies	187
11.3.2	Basics of DFT and DDG	188
11.3.3	DDG elements and characteristics	189
11.3.4	Information sources and generic procedure for DDG construction	191
11.3.5	Building DDG indirectly	192
11.3.6	Dealing with loops	194
11.4	DFT: Coverage and Applications	195
11.4.1	Achieving slice and other coverage	195
11.4.2	DFT: Applications and other issues	198
11.4.3	DFT application in synchronization testing	199
11.5	Concluding Remarks	200
	Problems	200
12	Testing Techniques: Adaptation, Specialization, and Integration	203
12.1	Testing Sub-Phases and Applicable Testing Techniques	203
12.2	Specialized Test Tasks and Techniques	210
12.3	Test Integration	214
12.4	Test Integration Example: Hierarchical Web Testing	214
12.5	Concluding Remarks	217
	Problems	219
PART III QUALITY ASSURANCE BEYOND TESTING		
13	Defect Prevention and Process Improvement	223
13.1	Basic Concepts and Generic Approaches	223

13.2 Education and Training for Defect Prevention	226
13.3 Other Techniques for Defect Prevention	228
13.3.1 Analysis and modeling for defect prevention	228
13.3.2 Technologies, standards, and methodologies for defect prevention	229
13.3.3 Software tools to block defect injection	230
13.4 Focusing on Software Processes	231
13.4.1 Process selection, definition, and conformance	232
13.4.2 Process maturity	232
13.4.3 Process and quality improvement	234
13.5 Concluding Remarks	234
Problems	235
14 Software Inspection	237
14.1 Basic Concepts and Generic Process	237
14.2 Fagan inspection	239
14.3 Other Inspections and Related Activities	242
14.3.1 Inspections of reduced scope or team size	242
14.3.2 Inspections of enlarged scope or team size	243
14.3.3 Informal desk checks, reviews, and walkthroughs	244
14.3.4 Code reading	245
14.3.5 Other formal reviews and static analyses	246
14.3.6 Tool support and process integration	247
14.4 Defect Detection Techniques and Inspection Effectiveness	247
14.5 Concluding Remarks	249
Problems	250
15 Formal Verification	251
15.1 Basic Concepts: Formal Verification and Formal Specification	251
15.2 Formal Verification: Axiomatic Approach	254
15.2.1 Formal logic specifications	254
15.2.2 Axioms	255
15.2.3 Axiomatic proofs and a comprehensive example	257
15.3 Other Approaches	259
15.3.1 Weakest pre-conditions and backward chaining	260
15.3.2 Functional approach and symbolic execution	260
15.3.3 General observations	261
15.3.4 Model checking and other approaches	262
15.4 Applications, Effectiveness, and Integration Issues	263
15.5 Concluding Remarks	265
Problems	266

16 Fault Tolerance and Failure Containment	267
16.1 Basic Ideas and Concepts	267
16.1.1 Ideas from other highly dependable systems	268
16.1.2 Adoption and adaptation to computers and software	269
16.1.3 Classification of techniques	269
16.2 Fault Tolerance with Recovery Blocks	270
16.3 Fault Tolerance with N-Version Programming	272
16.3.1 NVP: Basic technique and implementation	272
16.3.2 Ensuring version independence	273
16.3.3 Applying NVP ideas in other QA activities	274
16.4 Failure Containment: Safety Assurance and Damage Control	275
16.4.1 Hazard analysis using fault-trees and event-trees for accident prevention	275
16.4.2 Hazard resolution for accident prevention	277
16.4.3 Accident analysis and post-accident damage control	279
16.5 Application in Heterogeneous Systems	279
16.5.1 Modeling and analyzing heterogeneous systems	279
16.5.2 Prescriptive specifications for safety	281
16.6 Concluding Remarks	282
Problems	282
17 Comparing Quality Assurance Techniques and Activities	285
17.1 General Questions: Cost, Benefit, and Environment	285
17.2 Applicability to Different Environments	289
17.3 Effectiveness Comparison	291
17.3.1 Defect perspective	291
17.3.2 Problem types	292
17.3.3 Defect level and pervasiveness	293
17.3.4 Result interpretation and constructive information	294
17.4 Cost Comparison	295
17.5 Comparison Summary and Recommendations	297
Problems	298
PART IV QUANTIFIABLE QUALITY IMPROVEMENT	
18 Feedback Loop and Activities for Quantifiable Quality Improvement	303
18.1 QA Monitoring and Measurement	304
18.1.1 Direct vs indirect quality measurements	304
18.1.2 Direct quality measurements: Result and defect measurements	306
18.1.3 Indirect quality measurements: Environmental, product internal, and activity measurements	306
18.2 Immediate Followup Actions and Feedback	308
18.3 Analyses and Followup Actions	309

18.3.1	Analyses for product release decisions	309
18.3.2	Analyses for other project management decisions	311
18.3.3	Feedback to analyses and models themselves	312
18.3.4	Longer term and broader scope followup actions	313
18.4	Implementation, Integration, and Tool Support	314
18.4.1	Feedback loop: Implementation and integration	314
18.4.2	A refined quality engineering process	315
18.4.3	Tool support: Strategy, implementation, and integration	317
18.5	Concluding Remarks	320
	Problems	320
19	Quality Models and Measurements	323
19.1	Models for Quality Assessment	323
19.2	Generalized Models	324
19.3	Product-Specific Models	327
19.4	Model Comparison and Interconnections	328
19.5	Data Requirements and Measurement	330
19.6	Selecting Measurements and Models	333
19.7	Concluding Remarks	335
	Problems	337
20	Defect Classification and Analysis	339
20.1	General Types of Defect Analyses	339
20.1.1	Defect Distribution Analysis	340
20.1.2	Defect Trend Analysis and Defect Dynamics Model	343
20.1.3	Defect Causal Analysis	344
20.2	Defect Classification and ODC	345
20.2.1	ODC concepts	345
20.2.2	Defect classification using ODC: A comprehensive example	346
20.2.3	Adapting ODC to analyze web errors	347
20.3	Defect Analysis for Classified Data	348
20.3.1	One-way analysis: Analyzing a single defect attribute	348
20.3.2	Two-way and multi-way analysis: Examining cross-interactions	349
20.4	Concluding Remarks	350
	Problems	351
21	Risk Identification for Quantifiable Quality Improvement	353
21.1	Basic Ideas and Concepts	353
21.2	Traditional Statistical Analysis Techniques	355
21.3	New Techniques for Risk Identification	356
21.3.1	Principal component and discriminant analyses	356

21.3.2 Artificial neural networks and learning algorithms	358
21.3.3 Data partitions and tree-based modeling	359
21.3.4 Pattern matching and optimal set reduction	362
21.4 Comparisons and Integration	363
21.5 Risk Identification for Classified Defect Data	365
21.6 Concluding Remarks	368
Problems	369
22 Software Reliability Engineering	371
22.1 SRE: Basic Concepts and General Approaches	371
22.2 Large Software Systems and Reliability Analyses	372
22.3 Reliability Snapshots Using IDRM	374
22.4 Longer Term Reliability Analyses Using SRGMs	377
22.5 TBRMs for Reliability Analysis and Improvement	380
22.5.1 Constructing and using TBRMs	381
22.5.2 TBRM Applications	382
22.5.3 TBRM's impacts on reliability improvement	384
22.6 Implementation and Software Tool Support	385
22.7 SRE: Summary and Perspectives	386
Problems	387
Index	401

Bibliography

- Aldemir, T., Siu., N., Mosleh, A., Cacciabue, C., and Goktepe, P. G. (1994). *Reliability and Safety Assessment of Dynamic Process Systems*. NATO ASI Series. Springer-Verlag, New York.
- Allen, F. E. and Cocke, J. (1972). Graph theoretic constructs for program control flow analysis. Technical Report RC3923, IBM T. J. Watson Research Center.
- Avizienis, A. A. (1995). The methodology of N-version programming. In Lyu, M. R., editor, *Software Fault Tolerance*, pages 23–46. John Wiley & Sons, Inc., New York.
- Avritzer, A. and Weyuker, E. J. (1995). The automatic generation of load test suites and the assessment of the resulting software. *IEEE Trans. on Software Engineering*, 21(9):705–716.
- Bachiochi, D. J., Berstene, M. C., Chouinard, E. F., Conlan, N. M., Danchak, M. M., Furey, T., Neligon, C. A., and Way, D. (1997). Usability studies and designing navigational aids for the World Wide Web. *Computer Networks and ISDN Systems*, 29(8-13):1489–1496.
- Basili, V. R. (1995). The experience factory and its relationship to other quality approaches. In Zelkowitz, M. V., editor, *Advances in Computers, Vol.41*, pages 65–82. Academic Press, San Diego, CA.
- Basili, V. R. and Mills, H. D. (1982). Understanding and documenting programs. *IEEE Trans. on Software Engineering*, 8(3):270–283.

- Basili, V. R. and Rombach, H. D. (1988). The TAME project: Towards improvement-oriented software environments. *IEEE Trans. on Software Engineering*, 14(6):758–773.
- Basili, V. R., Zelkowitz, M. V., McGarry, F. E., Page, J., Waligora, S., and Pajerski, R. (1995). SEL's software process-improvement program. *IEEE Software*, 12(6):83–87.
- Beck, K. (1999). *Extreme Programming Explained: Embrace Change*. Addison-Wesley, Reading, Mass.
- Beck, K. (2003). *Test-Driven Development*. Addison-Wesley, Reading, Mass.
- Behlandorf, B. (1996). *Running a Perfect Web Site with Apache, 2nd Ed.* MacMillan Computer Publishing, New York.
- Beizer, B. (1990). *Software Testing Techniques, 2nd Ed.* International Thomson Computer Press, Boston, MA.
- Beizer, B. (1995). *Black-Box Testing: Techniques for Functional Testing of Software and Systems*. John Wiley & Sons, Inc., New York.
- Beizer, B. (1998). Software is different. *Software Quality Professional*, 1(1):44–54.
- Bhandari, I., Halliday, M., Tarver, E., Brown, D., Chaar, J., and Chillarege, R. (1993). A case study of software process improvement during development. *IEEE Trans. on Software Engineering*, 19(12):1157–1170.
- Biffi, S. and Halling, M. (2003). Investigating the defect detection effectiveness and cost benefit of nominal inspection teams. *IEEE Trans. on Software Engineering*, 29(5):385–397.
- Binder, R. V. (2000). *Testing Object Oriented Systems, Models, Patterns, and Tools*. Addison Wesley, Addison Wesley Longman Inc., One Jacob Way Reading, Massachusetts 01867.
- Bisant, D. B. and Lyle, J. R. (1989). A two-person inspection method to improve programming productivity. *IEEE Trans. on Software Engineering*, 15(10):1294–1304.
- Black, R. (2004). *Critical Testing Processes*. Addison-weley, Reading, MA.
- Blum, B. I. (1992). *Software Engineering: A Holistic View*. Oxford University Press, New York, NY.
- Boehm, B. and Basili, V. R. (2001). Software defect reduction top 10 list. *IEEE Computer*, 34(1):135–137.
- Boehm, B. W. (1981). *Software Engineering Economics*. Prentice Hall, Englewood Cliffs, New Jersey.
- Boehm, B. W. (1988). A spiral model of software development and enhancement. *IEEE Computer*, pages 61–72.
- Boehm, B. W. (1991). Software risk management: Principles and practices. *IEEE Software*, 8(1):32–41.

- Bowers, N. (1996). Weblint: Quality assurance for the World-Wide Web. *Computer Networks and ISDN Systems*, 28(7-11):1283–1290.
- Briand, L. C., Basili, V. R., and Hetmanski, C. J. (1993). Developing interpretable models with optimal set reduction for identifying high-risk software components. *IEEE Trans. on Software Engineering*, 19(11):1028–1044.
- Briand, L. C., Bunse, C., and Daly, J. W. (2001). A controlled experiment for evaluating quality guidelines on the maintainability of object-oriented designs. *IEEE Trans. on Software Engineering*, 27(6):513–530.
- Brooks, F. P. (1987). No silver bullet, essence and accidents of software engineering. *IEEE Computer*, 20(4):10–19.
- Brooks, F. P. (1995). *The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition*. Addison-Wesley Publishing Company, Reading, MA.
- Brown, J. R. and Lipow, M. (1975). Testing for software reliability. In *Proc. Int. Conf. Reliable Software*, pages 518–527, Los Sangeles, CA.
- Burnstein, I. (2003). *Practical Software Testing*. Springer-Verlag, New York.
- Buss, E. and Henshaw, J. (1992). Experiences in program understanding. Technical Report TR-74.105, IBM PRGS Toronto Laboratory.
- Card, D. N. and Glass, R. L. (1990). *Measuring Software Design Quality*. Prentice Hall, Englewood Cliffs, New Jersey.
- Cárdenas-García, S. R., Tian, J., and Zekowitz, M. V. (1992). An application of decision theory for the evaluation of software prototypes. *Journal of Systems and Software*, 19(1):27–39.
- Cárdenas-García, S. R. and Zekowitz, M. V. (1991). A management tool for evaluation of software designs. *IEEE Trans. on Software Engineering*, 17(9):961–971.
- Chaar, J., Halliday, M., Bhandari, I., and Chillarege, R. (1993). In-process evaluation for software inspection and test. *IEEE Trans. on Software Engineering*, 19(11):1055–1070.
- Charette, R. (1989). *Software Engineering Risk Analysis and Management*. McGraw Hill, New York.
- Chen, M. H., Lyu, M. R., and Wong, W. E. (2001). Effect of code coverage on software reliability measurement. *IEEE Trans. on Reliability*, 50(2):165–170.
- Chillarege, R., Bhandari, I., Chaar, J., Halliday, M., Moebus, D., Ray, B., and Wong, M.-Y. (1992). Orthogonal defect classification — a concept for in-process measurements. *IEEE Trans. on Software Engineering*, 18(11):943–956.
- Chow, T. S. (1978). Testing software design modeled by finite-state machines. *IEEE Trans. on Software Engineering*, 4(3):178–187.
- Chruscielski, K. and Tian, J. (1997). An operational profile for the cartridge support software. In *Proc. 8th Int. Symp. on Software Reliability Engineering*, pages 203–212.

- Clark, L. A. and Pregibon, D. (1993). Tree based models. In Chambers, J. M. and Hastie, T. J., editors, *Statistical Models in S*, chapter 9, pages 377–419. Chapman & Hall, London.
- Clarke, L. A. (1976). A system to generate test data and symbolically execute programs. *IEEE Trans. on Software Engineering*, 2(3):215–222.
- Clarke, L. A., Hassel, J., and Richardson, D. J. (1982). A close look at domain testing. *IEEE Trans. on Software Engineering*, 8:380–390.
- Cohen, E. I. (1978). *A Finite Domain-Testing Strategy for Computer Program Testing*. PhD thesis, Ohio State University.
- Denning, P. J. (1992). What is software quality? *Communications of the ACM*, 35(1):13–15.
- Deo, N. (1974). *Graph Theory with Applications to Engineering and Computer Science*. Prentice-Hall, Englewood Cliffs, New Jersey.
- Dijkstra, E. W. (1968). Go To statement considered harmful. *Communications of the ACM*, 11(3):147–148.
- Dijkstra, E. W. (1975). Guarded commands, nondeterminacy, and formal derivation of programs. *Communications of the ACM*, 18(8):453–457. EWD472.
- Dromey, R. G. (1995). A model for software product quality. *IEEE Trans. on Software Engineering*, 13(2):146–162.
- Dromey, R. G. (1996). Cornering the chimera. *IEEE Software*, 13(1):33–43.
- Dugan, J. B. (1995). Software system analysis using fault trees. In Lyu, M. R., editor, *Handbook of Software Reliability Engineering*, pages 615–659. McGraw-Hill, New York.
- Dunsmore, A., Roper, M., and Wood, M. (2003a). The development and evaluation of three diverse techniques for object-oriented code inspection. *IEEE Trans. on Software Engineering*, 29(8):677–686.
- Dunsmore, A., Roper, M., and Wood, M. (2003b). Practical code inspection techniques for object-oriented systems: An experimental comparison. *IEEE Software*, 20(4):21–29.
- Duran, J. W. and Ntafos, S. C. (1984). An evaluation of random testing. *IEEE Transactions On Software Engineering*, SE-10(4):438–444.
- Fagan, M. E. (1976). Design and code inspections to reduce errors in program development. *IBM Systems Journal*, 3:182–211.
- Farr, W. J. and Smith, O. D. (1991). Statistical modeling and estimation of reliability functions for software (SMERFS) users guide. Technical Report NSWC TR 84-373, Revision 2, Naval Surface Warfare Center.
- Fenton, N. and Pfleeger, S. L. (1996). *Software Metrics: A Rigorous and Practical Approach, 2nd Edition*. PWS Publishing.
- Frankl, P. G., Hamlet, R. G., Littlewood, B., and Strigini, L. (1998). Evaluating testing methods by delivered reliability. *IEEE Trans. on Software Engineering*, 24(8):586–601.

- Frankl, P. G. and Weyuker, E. J. (2000). Testing software to detect and reduce risk. *Journal of Systems and Software*, 53(3):275–286.
- Fromme, B. (1998). Web software testing: Challenges and solutions. In *InterWorks'98*.
- Garg, V. K. (1999). *IS-95 CDMA & CDMA 2000: Cellular/PCS Systems Implementation*. Prentice-Hall, Englewood Cliffs, New Jersey.
- Gerhart, S. A., Craigen, D., and Ralston, T. (1994). Experience with formal methods in critical systems. *IEEE Software*, 11(1):20–28.
- Ghezzi, C., Jazayeri, M., and Mandrioli, D. (2003). *Fundamentals of Software Engineering, 2nd Edition*. Prentice Hall, Englewood Cliffs, NJ.
- Gilb, T. and Graham, D. (1993). *Software Inspection*. Addison-Wesley Longman, London.
- Goel, A. L. (1985). Software reliability models: Assumptions, limitations, and applicability. *IEEE Trans. on Software Engineering*, 11(12):1411–1423.
- Goel, A. L. and Okumoto, K. (1979). A time dependent error detection rate model for software reliability and other performance measures. *IEEE Trans. on Reliability*, 28(3):206–211.
- Goodenough, J. B. and Gerhart, S. A. (1975). Toward a theory of test data selection. *IEEE Trans. on Software Engineering*, 1:156–173.
- Gries, D. (1987). *The Science of Programming*. Springer-Verlag.
- Gutttag, J. V., Horowitz, E., and Musser, D. R. (1978). Abstract data types and software validation. *Communications of the ACM*, 21(12).
- Hamlet, D., Mason, D., and Voit, D. (2001). Theory of software reliability based on components. In *Proc. 23rd Int. Conf. on Software Engineering*, pages 361–370, Toronto, Canada.
- Hamlet, D. and Taylor, R. (1990). Partition testing does not inspire confidence. *IEEE Trans. on Software Engineering*, 16(12):1402–1411.
- Hamlet, R. G. (1977). Testing programs with the aid of a compiler. *IEEE Trans. on Software Engineering*, 3:279–290.
- Hatton, L. (1998). Does OO sync with how we think? *IEEE Software*, 15(3):46–54.
- Henley, E. J. and Kumamoto, H. (1981). *Reliability Engineering and Risk Assessment*. Prentice-Hall, Englewood Cliffs, New Jersey.
- Hoare, C. A. R. (1969). An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580.
- Holmes, J. S. (2003). Identifying code-inspection improvements using statistical black belt techniques. *Software Quality Professional*, 6(1):4–14.
- Horgan, J. R. and Mathur, A. P. (1995). Software testing and reliability. In Lyu, M. R., editor, *Handbook of Software Reliability Engineering*, pages 531–566. McGraw-Hill, New York.

- Howden, W. E. (1976). Reliability of the path analysis testing strategy. *IEEE Trans. on Software Engineering*, 2(3):208–215.
- Howden, W. E. (1980). Functional testing. *IEEE Trans. on Software Engineering*, SE-6(2):162–169.
- Howden, W. E. (1982). Weak mutation testing and completeness of test sets. *IEEE Trans. on Software Engineering*, SE-8:371–379.
- Humphrey, W. (1998). The software quality profile. *Software Quality Professional*, 1(1):8–18.
- Humphrey, W. S. (1989). *Managing the Software Process*. Addison-Wesley, Reading, MA.
- Humphrey, W. S. (1995). *A Discipline for Software Engineering*. Addison-Wesley, Reading, MA.
- Huo, Q., Zhu, H., and Greenwood, S. (2003). A multi-agent software environment for testing web-based applications. In *Proc. 27th Int. Computer Software and Applications Conf.*, pages 210–215, Dallas, Texas.
- IBM (1991). *Programming Process Architecture, Version 2.1*. IBM.
- IEEE (1990). *IEEE Standard Glossary of Software Engineering Terminology*. Number STD 610.12-1990. IEEE.
- ISO (2001). *ISO/IEC 9126-1:2001 Software Engineering – Product Quality – Part 1: Quality Model*. ISO.
- Jain, A. K., Mao, J., and Mohiuddin, K. M. (1996). Artificial neural networks: A tutorial. *IEEE Computer*, 29(3):31–44.
- Jelinski, Z. and Moranda, P. L. (1972). Software reliability research. In Freiburger, W., editor, *Statistical Computer Performance Evaluation*, pages 365–484. Academic Press, New York.
- Jeng, B. and Weyuker, E. J. (1994). A simplified domain-testing strategy. *ACM Trans. on Software Engineering and Methodology*, 3(3):254–270.
- Kallepalli, C. and Tian, J. (2001). Measuring and modeling usage and reliability for statistical web testing. *IEEE Trans. on Software Engineering*, 27(11):1023–1036.
- Kan, S. H. (2002). *Metrics and Models in Software Quality Engineering, 2/e*. Addison-Wesley, Reading, MA.
- Kaner, C., Falk, J., and Nguyen, H. Q. (1999). *Testing Computer Software*. John Wiley & Sons, Inc., New York.
- Karlin, S. and Taylor, H. M. (1975). *A First Course in Stochastic Processes, 2nd Ed.* Academic Press, New York.
- Khoshgoftaar, T. M., Allen, E. B., Kalaichelvan, K. S., and Goel, N. (1996). Early quality prediction: A case study in telecommunications. *IEEE Software*, 13(1):65–71.
- Khoshgoftaar, T. M. and Szabo, R. M. (1996). Using neural networks to predict software faults during testing. *IEEE Trans. on Reliability*, 45(3):456–462.

- King, S., Hammond, J., Chapman, R., and Pryor, A. (2000). Is proof more cost-effective than testing. *IEEE Trans. on Software Engineering*, 26(8):675–686.
- Kitchenham, B. and Pfleeger, S. L. (1996). Software quality: The elusive target. *IEEE Software*, 13(1):12–21.
- Knight, J. C. and Myers, E. A. (1992). An improved inspection technique. *Communications of the ACM*, 36(11):51–61.
- Knuth, D. E. (1973). *The Art of Computer Programming*. Addison-Wesley, Reading, MA.
- Koru, A. G. and Tian, J. (2003). An empirical comparison and characterization of high defect and high complexity modules. *Journal of Systems and Software*, 67(3):153–163.
- Koru, A. G. and Tian, J. (2004). Defect handling in medium and large open source software projects. *IEEE Software*, 21(4):54–61.
- Krishnan, M. S. and Kellner, M. I. (1999). Measuring process consistency: Implications for reducing software defects. *IEEE Trans. on Software Engineering*, 25(6):800–815.
- Kung, D. C., Hsia, P., and Gao, J. (1998). *Testing Object-Oriented Software*. IEEE Computer Society Press, Los Alamitos, California.
- Kuvaja, P., Simila, J., Krzanik, L., Bicego, A., Koch, G., and Saukonen, S. (1994). *Software Process Assessment and Improvement: the BOOTSTRAP Approach*. Blackwell Publishers, Oxford, UK.
- Leveson, N. G. (1995). *Safeware: System Safety and Computers*. Addison-Wesley, Reading, MA.
- Li, Z. and Tian, J. (2003). Analyzing web logs to identify common errors and improve web reliability. In *Proc. IADIS International Conference on e-Society*, pages 235–242, Lisbon, Portugal.
- Lu, P. and Tian, J. (1993a). Applying software reliability engineering in large-scale software development. In *Proc. 3rd Int. Conf. on Software Quality*, pages 323–330, Lake Tahoe, Nevada.
- Lu, P. and Tian, J. (1993b). Software reliability engineering experience in the IBM Toronto Laboratory. In *Proc. IBM Software Engineering ITL Conf.*, pages 459–467, Toronto, Canada.
- Luqi (1989). Software evolution through rapid prototyping. *IEEE Computer*, pages 13–25.
- Lutz, R. R. and Mikulski, I. C. (2004). Ongoing requirements discovery in high-integrity systems. *IEEE Software*, 21(2):19–25.
- Lyu, M. R., editor (1995a). *Handbook of Software Reliability Engineering*. McGraw-Hill, New York.
- Lyu, M. R., editor (1995b). *Software Fault Tolerance*. John Wiley & Sons, Inc., New York.
- Lyu, M. R. and Avizienis, A. A. (1992). Assuring design diversity in N-version software: A design paradigm for N-version programming. In Meyer, J. F. and Schlichting, R. D.,

- editors, *Dependable Computing for Critical Applications 2*. Springer-Verlag, New York.
- Ma, L. and Tian, J. (2003). Analyzing errors and referral pairs to characterize common problems and improve web reliability. In *Proc. 3rd International Conference on Web Engineering*, pages 314–323, Oviedo, Spain.
- Mackenzie, D. (1994). Computer-related accidental death: An empirical exploration. *Science and Public Policy*, pages 233–248.
- Maddux, R. (1985). *A study of program structure*, Ph.D. dissertation. PhD thesis, University of Waterloo.
- Malaiya, Y. K., Li, M. N., Bieman, J. M., and Karcich, R. (2002). Software reliability growth with test coverage. *IEEE Trans. on Reliability*, 51(4):420–426.
- Malaiya, Y. K. and Srimani, P. K. (1990). *Software Reliability Models: Theoretical Developments, Evaluation & Applications*. IEEE Computer Society Press, Los Alamitos, California.
- DeMillo, R. A., McCracken, W. M., Martin, R. J., and Passafiume, J. F. (1987). *Software Testing and Evaluation*. Benjamin/Cummings, Menlo Park, CA.
- McAllister, D. F. and Vouk, M. A. (1995). Fault-tolerant software reliability engineering. In Lyu, M. R., editor, *Handbook of Software Reliability Engineering*, pages 567–614. McGraw-Hill, New York.
- McCabe, T. J. (1976). A complexity measure. *IEEE Trans. on Software Engineering*, 2(6):308–320.
- Mealy, G. H. (1955). A method for synthesizing sequential circuits. *Bell System Technical Journal*, 34:1045–1079.
- Miller, E. (2000). *The Website Quality Challenge*. Software Research Inc.
- Miller, E. F. and Howden, W. E. (1981). *Tutorial: Software Testing and Validation Techniques, 2nd Ed.* IEEE Computer Society.
- Mills, H. D. (1972). On the statistical validation of computer programs. Technical Report 72-6015, IBM Federal Syst. Div.
- Mills, H. D., Basili, V. R., Gannon, J. D., and Hamlet, R. G. (1987a). *Principles of Computer Programming: A Mathematical Approach*. Alan and Bacon, Inc., Boston, MA.
- Mills, H. D., Dyer, M., and Linger, R. C. (1987b). Cleanroom software engineering. *IEEE Software*, 4(5):19–24.
- Moore, E. F. (1956). Gedanken experiments on sequential machines. *Automata Studies. Annals of Mathematical Studies #34*.
- Munson, J. C. and Khoshgoftaar, T. M. (1992). The detection of fault-prone programs. *IEEE Trans. on Software Engineering*, 18(5):423–433.
- Musa, J. D. (1975). A theory of software reliability and its application. *IEEE Trans. on Software Engineering*, 1(3):312–327.

- Musa, J. D. (1993). Operational profiles in software reliability engineering. *IEEE Software*, 10(2):14–32.
- Musa, J. D. (1998). *Software Reliability Engineering*. McGraw-Hill, New York.
- Musa, J. D. and Everett, W. W. (1990). Software-reliability engineering: Technology for the 1990s. *IEEE Software*, 7(6):36–43.
- Musa, J. D., Iannino, A., and Okumoto, K. (1987). *Software Reliability: Measurement, Prediction, Application*. McGraw-Hill, New York.
- Musa, J. D. and Okumoto, K. (1984). A logarithmic Poisson execution time model for software reliability measurement. In *Proc. 7th Int. Conf. on Software Engineering*, pages 230–238, Orlando, FL.
- Myers, G. J. (1979). *The Art of Software Testing*. John Wiley & Sons, Inc., New York.
- Nelson, E. (1978). Estimating software reliability from test data. *Microelectronics and Reliability*, 17(1):67–73.
- Offutt, J. (2002). Quality attributes of web applications. *IEEE Software*, 19(2):25–32.
- Oivo, M. and Basili, V. R. (1992). Representing software engineering models: The TAME goal oriented approach. *IEEE Trans. on Software Engineering*, 18(10):886–898.
- Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058.
- Parnas, D. L. and Madey, J. (1995). Functional documentation for computer systems. *Sci. Comput. Program*, 25(1):41–61.
- Parnas, D. L. and Weiss, D. M. (1985). Active design reviews: Principles and practices. In *Proc. 8th Int. Conf. on Software Engineering*, pages 215–222. IEEE Computer Society Press.
- Paulk, M., Weber, C. V., Garcia, S. M., Chrissis, M. B., and Bush, M. W. (1993). Key practices of the capability maturity model, version 1.1. Technical Report CMU/SEI-93-TR-24, DTIC Number ADA263432, Software Engineering Institute.
- Paulk, M. C., Weber, C. V., Curtis, B., and Chrissis, M. B. (1995). *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison-Wesley, Reading, MA.
- Peterson, J. L. (1981). *Petri Net Theory and the Modeling of Systems*. Prentice Hall, Englewood Cliffs, New Jersey.
- Pfleeger, S. L. and Hatton, L. (1997). Investigating the influence of formal methods. *IEEE Computer*, 30(2):33–43.
- Pfleeger, S. L., Hatton, L., and Howell, C. C. (2002). *Solid Software*. Prentice Hall, Upper Saddle River, New Jersey.
- Porter, A. A. and Johnson, P. M. (1997). Assessing software review meetings: Results of a comparative analysis of two experimental studies. *IEEE Trans. on Software Engineering*, 23(3):129–145.

- Porter, A. A. and Selby, R. W. (1990). Empirically guided software development using metric-based classification trees. *IEEE Software*, 7(2):46–54.
- Porter, A. A., Siy, H., and Votta, L. G. (1996). A review of software inspections. In Zelkowitz, M. V., editor, *Advances in Computers, Vol.42*, pages 39–76. Academic Press, San Diego, CA.
- Porter, A. A. and Votta, L. G. (1997). What makes inspections work. *IEEE Software*, 14(5):99–102.
- Prahalad, C. K. and Krishnan, M. S. (1999). The new meaning of quality in the information age. *Harvard Business Review*, 77(5):109–118.
- Pratt, J. W., Raiffa, H., and Schlaifer, R. (1965). *Introduction to Statistical Decision Theory*. McGraw-Hill, New York.
- Pratt, T. W. and Zelkowitz, M. V. (2001). *Programming Languages: Design and Implementation*. Prentice-Hall, Inc., Upper Saddle River, New Jersey.
- Prechelt, L. (2000). An empirical comparison of seven programming languages. *IEEE Computer*, 33(10):23–29.
- Putnam, L. H. (1978). A general empirical solution to the macro software sizing and estimation problem. *IEEE Trans. on Software Engineering*, pages 345–361.
- Ramamoorthy, C. V. and Bastani, F. B. (1982). Software reliability: Status and perspectives. *IEEE Trans. on Software Engineering*, 8(4):359–371.
- Raymond, E. S. (1999). *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly and Associates, Sebastopol, CA, 95472, USA.
- Reichheld Jr., F. F. and Sasser, W. E. (1990). Zero defections: Quality comes to services. *Harvard Business Review*, 68(5):105–111.
- Rosenblum, D. S. and Weyuker, E. J. (1997). Using coverage information to predict the cost-effectiveness of regression testing strategies. *IEEE Trans. on Software Engineering*, 23(3):146–156.
- Rothermel, G. and Harrold, M. J. (1996). Analyzing regression test selection techniques. *IEEE Trans. on Software Engineering*, 22(8):529–551.
- Seaman, C. B. and Basili, V. R. (1997). Communication and organization in software development: An empirical study. *IBM Systems Journal*, 36.
- Seaman, C. B. and Basili, V. R. (1998). Communication and organization: An empirical study of discussion in inspection meetings. *IEEE Trans. on Software Engineering*, 24(7):559–572.
- Selby, R. W., Basili, V. R., and Baker, F. T. (1987). Cleanroom software development: An empirical evaluation. *IEEE Trans. on Software Engineering*, SE-13(9):1027–1037.
- Selby, R. W. and Porter, A. A. (1988). Learning from examples: Generation and evaluation of decision trees for software resource analysis. *IEEE Trans. on Software Engineering*, 14(12):1743–1757.

- Shneiderman, B. (1977). Measuring computer program quality and comprehension. *Int. J. of Man-Machine Studies*, 9.
- Shneiderman, B. (1980). *Software Psychology*. Winthrop Publishers, Cambridge, MA.
- Spiliopoulou, M. (2000). Web usage mining for web site evaluation. *Communications of the ACM*, 43(8):127–134.
- StatSci (1993). *S-PLUS Programmer's Manual, Version 3.2*. StatSci, A Division of Math-Soft, Inc., Seattle, Washington.
- Tai, K.-C. (1984). A program complexity metric based on data flow information in control graphs. In *7th Int. Conf. on Software Engineering*, pages 239–248, Orlando, Florida.
- Tanik, M. M. and Yeh, R. T. (1989). Rapid prototyping in software development. *IEEE Computer*, pages 9–10.
- Thayer, R., Lipow, M., and Nelson, E. (1978). *Software Reliability*. North-Holland.
- TIA/EIA (1994). *Mobile Station-Base Station Compatibility Standard for Dual Mode Wide-band Spread Spectrum Cellular System, Version 0.04*. TIA/EIA/IS-95-A.
- Tian, J. (1995). Integrating time domain and input domain analyses of software reliability using tree-based models. *IEEE Trans. on Software Engineering*, 21(12):945–958.
- Tian, J. (1996). An integrated approach to test tracking and analysis. *Journal of Systems and Software*, 35(2):127–140.
- Tian, J. (1998). Reliability measurement, analysis, and improvement for large software systems. In Zekowitz, M. V., editor, *Advances in Computers, Vol.46: The Engineering of Large Systems*, chapter 4, pages 159–235. Academic Press, San Diego, CA.
- Tian, J. (1999). Measurement and continuous improvement of software reliability throughout software life-cycle. *Journal of Systems and Software*, 47(2-3):189–195.
- Tian, J. (2000). Risk identification techniques for defect reduction and quality improvement. *Software Quality Professional*, 2(2):32–41.
- Tian, J. (2001). Quality assurance alternatives and techniques: A defect-based survey and analysis. *Software Quality Professional*, 3(3):6–18.
- Tian, J. (2002). Better reliability assessment and prediction through data clustering. *IEEE Trans. on Software Engineering*, 28(10):997–1007.
- Tian, J. (2004). Quality-evaluation models and measurements. *IEEE Software*, 21(3):84–91.
- Tian, J. and Henshaw, J. (1994). Tree-based defect analysis in testing. In *Proc. 4th Int. Conf. on Software Quality*, McLean, Virginia.
- Tian, J. and Lin, E. (1998). Unified Markov models for software testing, performance evaluation, and reliability analysis. In *4th ISSAT International Conference on Reliability and Quality in Design*, Seattle, Washington.
- Tian, J., Lu, P., and Palma, J. (1995). Test execution based reliability measurement and modeling for large commercial software. *IEEE Trans. on Software Engineering*, 21(5):405–414.

- Tian, J., Ma, L., Li, Z., and Koru, A. G. (2003). A hierarchical strategy for testing web-based applications and ensuring their reliability. In *Proc. 27th Int. Computer Software and Applications Conf. (1st IEEE Workshop on Web-based Systems and Applications)*, pages 702–707, Dallas, Texas.
- Tian, J. and Nguyen, A. (1999). Statistical web testing and reliability analysis. In *Proc. 9th Int. Conf. on Software Quality*, pages 263–274, Cambridge, MA.
- Tian, J., Nguyen, A., Allen, C., and Appan, R. (2001). Experience with identifying and characterizing problem prone modules in telecommunication software systems. *Journal of Systems and Software*, 57(3):207–215.
- Tian, J. and Palma, J. (1997). Test workload measurement and reliability analysis for large commercial software systems. *Annals of Software Engineering*, 4:201–222.
- Tian, J. and Palma, J. (1998). Analyzing and improving reliability: A tree based approach. *IEEE Software*, 15(2):97–104.
- Tian, J. and Troster, J. (1998). A comparison of measurement and defect characteristics of new and legacy software systems. *Journal of Systems and Software*, 44(2):135–146.
- Tian, J., Troster, J., and Palma, J. (1997). Tool support for software measurement, analysis, and improvement. *Journal of Systems and Software*, 39(2):165–178.
- Trivedi, K. S. (2001). *Probability and Statistics with Reliability, Queuing, and Computer Science Applications, 2nd Edition*. John Wiley & Sons, Inc., New York.
- Troster, J. and Tian, J. (1995). Measurement and defect modeling for a legacy software system. *Annals of Software Engineering*, 1:95–118.
- Troster, J. and Tian, J. (1996). Exploratory analysis tools for tree-based models in software measurement and analysis. In *Proc. 4th Int'l Symp. on Assessment of Software Tools*, pages 7–17, Toronto, Ontario, Canada.
- Tsoukalas, M. Z., Duran, J. W., and Ntafos, S. C. (1993). On some reliability estimation problems in random and partition testing. *IEEE Trans. on Software Engineering*, 19(7):687–697.
- van Solingen, R. and Berghout, E. (1999). *The Goal/Question/Metric Method: A Practical Method for Quality Improvement of Software Development*. McGraw-Hill.
- Vatanasombut, B., Stylianou, A. C., and Igarria, M. (2004). How to retain online customers. *Communications of the ACM*, 47(6):65–69.
- Venables, W. N. and Ripley, B. D. (1994). *Modern Applied Statistics with S-Plus*. Springer-Verlag, New York.
- Vixie, P. (1999). *Open Sources: Voices from the Open Source Revolution*, chapter Software Engineering, pages 91–100. O'Reilly & Associates, Inc, Sebastopol, CA, 95472, USA.
- Voas, J. (1998). *Software Fault Injection - Inoculating Programs Against Errors*. Wiley Computer Publishing.
- Voas, J. M. (1999). Certifying software for high-assurance environments. *IEEE Software*, 16(4):48–54.

- Voas, J. M. (2000). Developing a usage-based software certification process. *IEEE Computer*, 16(8):32–37.
- von Mayrhauser, A. (1990). *Software Engineering: Methods and Management*. Academic Press, San Diego, CA.
- Wallace, D. R., Ippolito, L. M., and Cuthill, B. (1996). *Reference Information for the Software Verification and Validation Process*. Number NIST Special Publication 500-234. NIST.
- Weiser, M. D. (1984). Program slicing. *IEEE Trans. on Software Engineering*, 10:352–357.
- Weyuker, E. J. (1998). Testing component-based software: A cautionary tale. *IEEE Software*, 15(5):54–59.
- Weyuker, E. J. and Jeng, B. (1991). Analyzing partition test strategies. *IEEE Trans. on Software Engineering*, 17(7):703–711.
- Weyuker, E. J., Ostrand, T. J., Brophy, J., and Prasad, R. (2000). Clearing a career path for software testers. *IEEE Software*, 17(2):76–82.
- White, L. J. and Cohen, E. I. (1980). A domain strategy for computer program testing. *IEEE Trans. on Software Engineering*, 6:247–257.
- Whittaker, J. A. (2001). Software’s invisible users. *IEEE Software*, 18(3):84–88.
- Whittaker, J. A. and Poore, J. H. (1993). Markov analysis of software specifications. *ACM Trans. on Software Engineering and Methodology*, 2(1):93–106.
- Whittaker, J. A. and Thomason, M. G. (1994). A Markov chain model for statistical software testing. *IEEE Trans. on Software Engineering*, 20(10):812–824.
- Wiener, R. (1998). Watch your language! *IEEE Software*, 15(3):55–56.
- Wirth, N. (1995). A plea for lean software. *IEEE Computer*, 28(2):64–68.
- Yamada, S., Ohba, M., and Osaki, S. (1983). S-shaped reliability growth modeling for software error detection. *IEEE Trans. on Reliability*, 32(5):475–478.
- Yih, S. and Tian, J. (1998). Developing and checking prescriptive specifications for safety improvement. *Microprocessors and Microsystems*, 21(10):587–594.
- Zelkowitz, M. V. (1988). Resource utilization during software development. *Journal of Systems and Software*, 8:331–336.
- Zelkowitz, M. V. (1993). Role of verification in the software specification process. In Yovits, M. C., editor, *Advances in Computers, Vol.36*, pages 43–109. Academic Press, San Diego, CA.
- Zhao, L. and Elbaum, S. (2003). Quality assurance under the open source development model. *Journal of Systems and Software*, 66(1):65–75.