Agrawal, H., and Horgan, J.R. (1990). "Dynamic Program Slicing," in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, 25(6), *ACM SIGPLAN Notices*, pp. 246–256, White Plains, NY.

Aho, A.V., Sethi, R., and Ullman, J.D. (1986). *Compilers—Principles, Techniques and Tools*, Addison-Wesley.

Ammons, G., Bodik, R., and Larus, J.R. (2002). "Mining Specifications," in *Proceedings of the ACM SIGPLAN/SIGACT Symposium on Principles of Programming Languages*, Portland, OR.

Artzi, S., Kim, S., and Ernst, M.D. (2008). "ReCrash: Making Software Failures Reproducible by Preserving Object States," in *Proceesings of ECOOP 2008, Object-Oriented Programming, 22nd European Conference,* Paphos, Cyprus.

Barnett, M., DeLine, R., Fähndrich, M., Leino, K.R.M., and Schulte, W. (2004). "Verification of Object-Oriented Programs with Invariants," *Journal of Object Technology* 3(6):27–56. Special Issue: ECOOP 2003 Workshop on Formal Techniques for Java-like Programs.

Barron, C.A. (2002). "High Tech's Missionaries of Sloppiness," *Salon Magazine*, December 2000.

Barrow, H.G., and Burstall, R.M. (1976). "Subgraph Isomorphism, Matching Relational Structures and Maximal Cliques," *Information Processing Letters* 4(4):83–84.

Beizer, B. (1990). *Software Testing Techniques*. International Thomson Computer Press.

Beizer, B. (1999). "Unbanning the 'Bug.'" Posting *79q48lncc1@fir.prod.itd.earthblink.net to comp.software.testing*.

Beizer, B. (2000). "Definition of the Word *Bug*." Posting *8kcV4.4008$S31.103769 @newsread2.prod.itd.earthblink.net to comp.software.testing.*

Beveridge, W.I.B. (1957). *The Art of Scientific Investigation*, 3rd ed., Vintage Books.

Binkley, D., and Harman, M. (2003). "A Large-Scale Empirical Study of Forward and Backward Static Slice Size and Context Sensitivity," in *Proceedings of the International Conference on Software Maintenance*, IEEE Computer Society.

Bloch, A., ed. (1980). *Murphy's Law Book Two: More Reasons Why Things Go Wrong!* Price/Stern/Sloan Publishers.

Booch, G. (1994). *Object-Oriented Analysis and Design*, 2nd ed., Addison-Wesley.

Bron, C., and Kerbosch, J. (1973). "Algorithm 457—Finding All Cliques of an Undirected Graph," *Communications of the ACM* 16(9):575–577.

Brun, Y., and Ernst, M. (2004). "Finding Latent Code Errors Via Machine Learning over Program Executions," in *Proceedings of the International Conference on Software Engineering*, pp. 480–490, Edinburgh.

Burdy, L., Cheon, Y., Cok, D., Ernst, M., Kiniry, J., Leavens, G.T., Leino, K.R.M., and Poll, E. (2003). "An Overview of JML Tools and Applications," in *Proceedings of the Eighth International Workshop on Formal Methods for Industrial Critical Systems*, Trondheim, Norway.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. (1996). *Pattern-Oriented Software Architecture: A System of Patterns*, volume 1, *Pattern-Oriented Software Architecture*, John Wiley & Sons.

Chaki, S., Groce, A., and Strichman, O. (2004). "Explaining Abstract Counterexamples," in *Proceedings of the 12th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 73–82, Newport Beach, CA.

Chelf, B. (2004). "Squashing Bugs at the Source," *Linux Magazine* 55:16–20.

Choi, J.-D., and Srinivasan, H. (1998). "Deterministic Replay of Java Multithreaded Applications," in *Proceedings of the ACM SIGMETRICS Symposium on Parallel and Distributed Tools*, pp. 48–59.

Choi, J.-D., and Zeller, A. (2002). "Isolating Failure-Inducing Thread Schedules," in *Proceedings of the International Symposium on Software Testing and Analysis*, pp. 201–220, Rome.

Cleve, H., and Zeller, A. (2005). "Locating Causes of Program Failures," in *Proceedings of the International Conference on Software Engineering*, St. Louis, MO.

Cohn, R., and Muth, R. (2004). *Pin 2.0 User Guide*. Available at: *http://rogue.colorado.edu/Pin/documentation.php.*

Condit, J., Harren, M., McPeak, S., Necula, G.C., and Weimer, W. (2003). "Cured in the Real World," in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 232–244, San Diego.

Dallmeier, V., Lindig, C., and Zeller, A. (2005). "Lightweight Defect Localization for Java," in *Proceedings of the 19th European Conference on Object-Oriented Programming*, Glasgow.

DeMillo, R.A., Pan, H., and Spafford, E.H. (1996). "Critical Slicing for Software Fault Localization," in *Proceedings of the International Symposium on Software Testing and Analysis*, pp. 121–134.

Demsky, B., and Rinard, M. (2003). "Automatic Detection and Repair of Errors in Data Structures," in *Proceedings of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pp. 78–95, Anaheim, CA.

Dickinson, W., Leon, D., and Podgurski, A. (2001). "Finding Failures by Cluster Analysis of Execution Profiles," in *Proceedings of the International Conference on Software Engineering*, pp. 339–348, Toronto.

Dijkstra, E.W. (1972). "Notes on Structured Programming," in Dahl, O.-J., Dijkstra, E.W., and Hoare, C.A.R., eds., *Structured Programming*, pp. 1–82, Academic Press.

Dijkstra, E.W. (1982). "On Webster, Users, Bugs, and Aristotle," in *Selected Writings on Computing: A Personal Perspective*, pp. 288–291, Springer-Verlag. (Originally published as EWD 618 in 1977.)

Dijkstra, E.W. (1989). "On the Cruelty of Really Teaching Computer Science," *Communications of the ACM* 32(12):1398–1404.

Ducassé, M. (1999). "Coca: An Automated Debugger for C," in *Proceedings of the International Conference on Software Engineering*, pp. 504–513, Los Angeles.

Dunlap, G.W., King, S.T., Cinar, S., Basrai, M.A., and Chen, P.M. (2002). "Revirt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay," in *Proceedings of the Symposium on Operating Systems Design and Implementation*, Boston.

Dustin, E., Rashka, J., and Paul, J. (2001). *Automated Software Testing: Introduction, Management, and Performance*, Addison-Wesley.

Eisenstadt, M. (1997). "My Hairiest Bug War Stories," *Communications of the ACM* 40(4): 30–37.

Endres, A., and Rombach, D. (2003). *A Handbook of Software and Systems Engineering*, Pearson/Addison-Wesley.

Ernst, M.D., Cockrell, J., Griswold, W.G., and Notkin, D. (2001). "Dynamically Discovering Likely Program Invariants to Support Program Evolution," *IEEE Transactions on Software Engineering* 27(2):1–25.

ESEC/FSE 99 (1999). *Proceedings of ESEC/FSE'99—7th European Software Engineering Conference/7th ACM SIGSOFT Symposium on the Foundations of Software Engineering*, volume 1687, *Lecture Notes in Computer Science*. Toulouse, France.

Fewster, M., and Graham, D. (1998). *Software Test Automation*, Addison-Wesley.

Fishman, C. (1996). "They Write the Right Stuff," *Fast Company Magazine* 6.

Fritzson, P., Shahmehri, N., Kamkar, M., and Gyimothy, T. (1992). "Generalized Algorithmic Debugging and Testing," *ACM Letters on Programming Languages and Systems* 1(4): 303–322.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley.

Geiger, L., and Zündorf, A. (2002). "Graph-Based Debugging with Fujaba," in *Workshop on Graph Based Tools, International Conference on Graph Transformations*, Barcelona.

Gould, J.D. (1975). "Some Psychological Evidence on How People Debug Computer Programs," *International Journal of Man-Machine Studies* 7:151–182.

Graves, T.L., Karr, A.F., Marron, J.S., and Siy, H. (2000). "Predicting Fault Incidence Using Software Change History," *IEEE Transactions on Software Engineering* 26(7):653–661.

Groce, A., and Visser, W. (2003). "What Went Wrong: Explaining Counterexamples," in *Proceedings of the SPIN Workshop on Model Checking of Software*, pp. 121–135, Portland, OR.

Gyimóthy, T., Beszédes, Á., and Forgács, I. (1999). "An Efficient Relevant Slicing Method for Debugging," in *ESEC/FSE' 99*, pp. 303–321.

Hailpern, B., and Santhanam, P. (2002). "Software Debugging, Testing, and Verification," *IBM Systems Journal* 41(1):4–12.

Hangal, S., and Lam, M.S. (2002). "Tracking Down Software Bugs Using Automatic Anomaly Detection," in *ICSE-2002*, pp. 291–302.

Hopper, G.M. (1981). "The First Bug," *Annals of the History of Computing* 3(3): 285–286.

Hovemeyer, D., and Pugh, W. (2004). "Finding Bugs Is Easy," in *Proceedings of the Conference on Object-Oriented Programming Systems Languages and Applications*, pp. 132–136, Vancouver, BC.

Hume, D. (1748). *An Enquiry Concerning Human Understanding*, A. Millar.

Humphrey, W.S. (1996). *Introduction to the Personal Software Process*, Addison-Wesley.

Humphrey, W.S. (1999). "Bugs or Defects?" Technical Report, Volume 2, Issue 1. Carnegie Mellon Software Engineering Institute.

ICSE (2002). *Proceedings of the International Conference on Software Engineering*, Orlando, FL.

Jacky, J. (1996). *The Way of Z: Practical Programming with Formal Methods*, Cambridge University Press.

Jim, T., Morrisett, J.G., Grossman, D., Hicks, M.W., Cheney, J., and Wang, Y. (2002). "Cyclone: A Safe Dialect of C," in *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, pp. 275–288.

Jones, J.A., Harrold, M.J., and Stasko, J. (2002). "Visualization of Test Information to Assist Fault Localization," in *ICSE 2002*, pp. 467–477.

Kaner, C., Falk, J., and Nguyen, H.Q. (1999). *Testing Computer Software*, John Wiley & Sons.

Kernighan, B.W., and Pike, R. (1999). *The Practice of Programming*, Addison-Wesley.

Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., and Griswold, W.G. (2001). "An Overview of AspectJ," in *Proceedings of the 15th European Conference on Object-Oriented Programming*, pp. 327–353, Budapest.

Kidder, T. (1981). *The Soul of a New Machine*. New York: Atlantic Monthly Press.

Kim, S., Zimmermann, T., Whitehead Jr., E.J., and Zeller, A. (2007). "Predicting Faults from Cached History," in *Proceedings of the 29th International Conference on Software Engineering*, pp. 489–498, Washington, DC.

Knight, J.C., and Leveson, N.G. (1986). "An Experimental Evaluation of the Assumption of Independence in Multiversion Programming," *IEEE Transactions on Software Engineering* 12(1):96–109.

Ko, A.J., and Myers, B.A. (2004). "Designing the Whyline: A Debugging Interface for Asking Questions about Program Behavior," in *CHI '04: Proceedings of the 2004 Conference on Human Factors in Computing Systems*, pp. 151–158, Vienna.

Ko, A.J., and Myers, B.A. (2005). "A Framework and Methodology for Studying the Causes of Software Errors in Programming Systems," *Journal of Visual Languages and Computing* 16(1–2):41–84.

Ko, A. J., and Myers, B.A. (2008). "Debugging Reinvented: Asking and Answering Why and Why Not Questions about Program Behavior," in *Proceedings of the 30th International Conference on Software Engineering*, Leipzig, Germany.

Ko, A. J., and Myers, B.A. (2009). "Finding Causes of Program Output with the Java Whyline," in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, Boston.

Kolawa, A. (2002). "Using Bug-Tracking Systems as Idea Repositories." Available at: *www.stickyminds.com*.

Konuru, R., Srinivasan, H., and Choi, J.-D. (2000). "Deterministic Replay of Distributed Java Applications," in *Proceedings of the International Parallel and Distributed Processing Symposium*, Cancun.

Korel, B., and Laski, J. (1990). "Dynamic Slicing of Computer Programs," *The Journal of Systems and Software* 13(3):187–195.

Larman, C. (2002). *Applying UML and Patterns*, Prentice Hall.

Leavens, G.T., Baker, A.L., and Ruby, C. (1999). "JML: A Notation for Detailed Design," in *Behavioral Specifications of Businesses and Systems*, pp. 175–188, Kluwer Academic Publishers.

Leavens, G.T., and Cheon, Y. (2004). "Design by Contract with JML," Technical Report, Iowa State University. Available at: *http://www.jmlspecs.org/*.

Leitner, A., Ciupa, I., Oriol, M., Meyer, B., and Fiva, A. (2007) "Contract Driven Development = Test Driven Development–Writing Test Cases," in *Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Dubrovnik, Croatia.

Lencevicius, R. (2000). *Advanced Debugging Methods*, Kluwer Academic Publishers.

Leveson, N.G., Cha, S.S., Knight, J.C., and Shimeall, T.J. (1990). "The Use of Self-Checks and Voting in Software Error Detection: An Empirical Study," *IEEE Transactions on Software Engineering* 16(4):432–443.

Lewis, B. (2003). "Debugging Backward in Time," in Ronsse, M., ed., *Proceedings of the Fifth International Workshop on Automated and Algorithmic Debugging*, Ghent, Belgium.

Lewis, D. (1973). "Causation," *Journal of Philosophy* 70:556–567. Reprinted in Lewis (1986).

Lewis, D. (1986). *Philosophical Papers: Volume II*, Oxford University Press.

Liblit, B., Aiken, A., Zheng, A.X., and Jordan, M.I. (2003). "Bug Isolation Via Remote Program Sampling," in *Proceedings of the SIGPLAN Conference on Programming Language Design and Implementation*, San Diego.

Liblit, B., Naik, M., Zheng, A.X., Aiken, A., and Jordan, M.I. (2005). "Scalable Statistical Bug Isolation," in *Proceedings of the SIGPLAN Conference on Programming Language Design and Implementation*, Chicago.

Martin, R.C. (1996). "The Dependency Inversion Principle," *C++ Report*, May 8, 1996.

McConnell, S.C. (1993). *Code Complete: A Practical Handbook of Software Construction*, Microsoft Press.

Meyer, B. (1997). *Object-Oriented Software Construction*, 2nd ed., Prentice-Hall.

Miller, B.P., Fredrikson, L., and So, B. (1990). "An Empirical Study of the Reliability of UNIX Utilities," *Communications of the ACM* 33(12):32–44.

Mirrer, B. (2000). "Organize Your Problem Tracking System," *Software Testing & Quality Engineering Magazine* 2(5).

Mockus, A., and Votta, L. G. (2000). "Identifying Reasons for Software Changes Using Historic Databases," in *Proceedings of the International Conference on Software Maintenance*, pp. 120–130, San Jose.

Mockus, A., and Weiss, D.M. (2000). "Predicting Risk of Software Changes," *Bell Labs Technical Journal* 5(2):169–180.

Morgenstern, C. (1964). "The Impossible Fact," in Knight, M., ed., *The Gallows Songs*. University of California Press. (Original poem published in 1905.)

Muchnik, S.S. (1997). *Advanced Compiler Design and Implementation*, Morgan Kaufmann.

Müller, M.M., Typke, R., and Hagner, O. (2002). "Two Controlled Experiments Concerning the Usefulness of Assertions as a Means for Programming," in *Proceedings of the 18th International Conference on Software Maintenance*, pp. 84–92, San Jose.

Myers, G.J. (1979). *The Art of Software Testing*, John Wiley & Sons.

Nagappan, N., and Ball, T. (2005). "Use of Relative Code Churn Measures to Predict System Defect Density," in *Proceedings of the 27th International Conference on Software Engineering*, St. Louis, MO.

Nagappan, N., Ball, T., and Zeller, A. (2006). "Mining Metrics to Predict Component Failures," in *Proceedings of the 28th International Conference on Software Engineering*, pp. 452–461, Shong Hai.

Nagappan, N., Zeller, A., and Zimmermann, T. (2008). "Predicting Bugs from History," in Tom Mens and Serge Demeyer, eds., *Software Evolution*, pp. 69–88, Springer.

Nagappan, N., Zeller, A., and Zimmermann, T., eds. (2009). *IEEE Software—Special Issue on Mining Software Repositories*, January.

Naish, L. (1997). "A Declarative Debugging Scheme," *The Journal of Functional and Logic Programming* 3.

Necula, G.C., McPeak, S., and Weimer, W. (2002). "Cured: Type-Safe Retrofitting of Legacy Code," in *Proceedings of the 29th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 128–139, Portland, OR.

Ness, B., and Ngo, V. (1997). "Regression Containment through Source Code Isolation," in *Proceedings of the 21st Annual International Computer Software & Applications Conference*, pp. 616–621, Washington, DC.

Nethercote, N. (2004). "Dynamic Binary Analysis and Instrumentation," Ph.D. Thesis, University of Cambridge, UK.

Nethercote, N., and Seward, J. (2003). "Valgrind: A Program Supervision Framework," *Electronic Notes in Theoretical Computer Science* 89(2).

Neuburg, M. (2003). *AppleScript: The Definitive Guide*, O'Reilly.

Neuhaus, S., Zimmermann, T., Holler, C., and Zeller, A. (2007). "Predicting Vulnerable Software Components," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, Alexandria, VA.

Ogawa, M. (2007). "Visualizing the Eclipse Bug Data." Available at: *http://vis.cs.ucdavis.edu/ogawa/eclipse/*.

Orso, A., Apiwattanapong, T., and Harrold, M.J. (2003). "Leveraging Field Data for Impact Analysis and Regression Testing," in *Proceedings of the 9th European Software Engineering Conference Held Jointly with 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 128–137.

Ostrand, T., and Weyuker, E. (2002). "The Distribution of Faults in a Large Industrial Software System," in Frankl, P.G., ed., *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis*, Volume 27(4), *Software Engineering Notes*, pp. 55–64.

Ostrand, T. J., Weyuker, E.J., and Bell, R.M. (2004). "Where the Bugs Are," in *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 86–96.

Ottenstein, K.J., and Ottenstein, L.M. (1984). "The Program Dependence Graph in a Software Development Environment," in *Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments*, Volume 19, *ACM SIGPLAN Notices*, pp. 177–184.

Pearl, J. (2000). *Causality: Models, Reasoning, and Inference*, Cambridge University Press.

Pezzè, M., and Young, M. (2005). *Software Testing and Analysis: Process, Principles, and Techniques*, John Wiley & Sons.

Pirsig, R.M. (1974). *Zen and the Art of Motorcycle Maintenance*, William Morrow.

Podgurski, A., Leon, D., Francis, P., Masri, W., Minch, M., Sun, J., and Wang, B. (2003). "Automated Support for Classifying Software Failure Reports," in *Proceedings of the 25th International Conference on Software Engineering*, pp. 465–475, Portland, OR.

Popper, K. (1959). *The Logic of Scientific Discovery*, Hutchinson. (Translation of *Logik der Forschung*, Vienna, 1935.)

Purushothaman, R., and Perry, D.E. (2004). "Towards Understanding the Rhetoric of Small Changes," in *Proceedings of the International Workshop on Mining Software Repositories*, pp. 90–94, Edinburgh.

Raymond, E.S., ed. (1996). *New Hacker's Dictionary*, 3rd ed., MIT Press. See also *http://www.jargon.org/*.

Renieris, M., and Reiss, S.P. (2003). "Fault Localization with Nearest Neighbor Queries," in *Proceedings of the 18th International Conference on Automated Software Engineering*, Montreal.

Ronsse, M., Bosschere, K.D., Christiaens, M., de Kergommeaux, J.C., and Kranzlmüller, D. (2003). "Record/Replay for Nondeterministic Program Executions," *Communications of the ACM* 46(9):62–67.

Rosenberg, J.B. (1996). *How Debuggers Work—Algorithms, Data Structures, and Architecture*, John Wiley & Sons.

RTI (2002). "The Economic Impacts of Inadequate Infrastructure for Software Testing," Technical Report, Planning Report 02-3, National Institute of Standards and Technology.

Saff, D., and Ernst, M. (2004a). "Automatic Mock Object Creation for Test Factoring," in Flanagan, C. and Zeller, A., eds., *Proceedings of the ACM SIGPLAN/SIGSOFT Workshop on Program Analysis for Software Tools and Engineering*, Washington, DC.

Saff, D., and Ernst, M. (2004b). "An Experimental Evaluation of Continuous Testing During Development," in *Proceedings of the International Symposium on Software Testing and Analysis*, pp. 76–85, Boston.

Schmidt, D.C., Stal, M., Rohnert, H., and Buschmann, F. (2000). *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects*, Volume 2, *Pattern-Oriented Software Architecture*, John Wiley & Sons, New York.

Shapiro, E.Y. (1982). "Algorithmic Program Debugging," Ph.D. Thesis, MIT Press, ACM Distinguished Dissertation.

Shapiro, F.R. (1994). "Exposing the Myth Behind the First Bug Reveals a Few Tales," *BYTE*.

Shore, J. (2004). "Fail Fast," *IEEE Software* 21(5):21–25.

Śliwerski, J., Zimmermann, T., and Zeller, A. (2005a). "When Do Changes Induce Fixes?" in *Proceedings of the Workshop on Mining Software Repositories*, St. Louis, MO.

Śliwerski, J., Zimmermann, T., and Zeller, A. (2005b). "HATARI: Raising Risk Awareness," in *Proceedings of the 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, Lisbon.

Sommerville, I. (2001). *Software Engineering*, 6th ed., Addison-Wesley.

Sosič, R., and Abramson, D. (1997). "Guard: A Relative Debugger," *Software—Practice and Experience* 27(2):185–106.

Stallman, R.M., and Pesch, R.H. (1994). *Debugging with GDB*, 4th ed., Free Software Foundation; distributed with GDB 4.13.

Tip, F. (1995). "A Survey of Program Slicing Techniques," *Journal of Programming Languages* 3(3):121–189.

Viega, J., and McGraw, G. (2001). *Building Secure Software*, Addison-Wesley.

Voas, J.M. (1992). "PIE: A Dynamic Failure-based Technique," *IEEE Transactions on Software Engineering* 18(8):717–727.

Wahbe, R. (1992). "Efficient Data Breakpoints," in *Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 200–212.

Wattenberg, M. (1998). "Map of the Market." Available at: *http://www.bewitched.com/*.

Weinberg, G.M. (1971). *The Psychology of Computer Programming*, Van Nostrand Reinhold.

Weiser, M. (1982). "Programmers Use Slices When Debugging," *Communications of the ACM* 25(7):446–452.

Weiser, M. (1984). "Program Slicing," *IEEE Transactions on Software Engineering* 10(4): 352–357.

Whalley, D.B. (1994). "Automatic Isolation of Compiler Errors," *ACM Transactions on Programming Languages and Systems* 16(5):1648–1659.

Wilson, E.B. (1952). *An Introduction to Scientific Research*, McGraw-Hill.

Xie, Y., and Engler, D. (2002). "Using Redundancies to Find Errors," in *Proceedings of the 10th ACM SIGSOFT Symposium on Foundations of Software Engineering*, pp. 51–60.

Zachary, G.P. (1994). *Showstopper!: The Breakneck Race to Create Windows NT and the Next Generation at Microsoft*, Free Press.

Zalta, E.N., ed. (2002). *Stanford Encyclopedia of Philosophy*. Stanford University. Available at: *http://plato.stanford.edu/*.

Zeller, A. (1999). "Yesterday, My Program Worked. Today, It Does Not. Why?" in *ESEC/FSE 99*, pp. 253–267.

Zeller, A. (2000). *Debugging with DDD*, Version 3.2, Universität Passau and Free Software Foundation; distributed with GNU DDD.

Zeller, A. (2002). "Isolating Cause–Effect Chains from Computer Programs," in Griswold, W. G., ed., *Proceedings of the Tenth ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pp. 1–10, Charleston, SC.

Zeller, A., and Hildebrandt, R. (2002). "Simplifying and Isolating Failure-Inducing Input," *IEEE Transactions on Software Engineering* 28(2):183–200.

Zeller, A., and Lütkehaus, D. (1996). "DDD—A Free Graphical Front-end for UNIX Debuggers," *SIGPLAN Notices* 31(1):22–27.

Zhang, X., and Gupta, R. (2004). "Cost-Effective Dynamic Program Slicing," in *Proceedings of the 2004 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pp. 94–106, Washington, DC.

Zimmermann, T., and Nagappan, N. (2008). "Predicting Defects Using Network Analysis on Dependency Graphs," in *Proceedings of the 30th International Conference on Software Engineering*, pp.531–540, Leipzig, Germany.

Zimmermann, T., Premraj, R., and Zeller, A. (2007). "Predicting Defects for ECLIPSE," in *Proceedings of the Third International Workshop on Predictor Models in Software Engineering*, p. 9, Minneapolis, MN.

Zimmermann, T., and Zeller, A. (2002). "Visualizing Memory Graphs," in Diehl, S., ed., *Proceedings of the International Dagstuhl Seminar on Software Visualization*, Volume 2269, *Lecture Notes in Computer Science*, pp. 191–204, Dagstuhl, Germany.