CURRICULUM VITAE

RICHARD GRAHAM HAMLET

October, 2007

Education

Degree	*Institution*	*Year*	*Specialty*
Ph.D.	University of Washington	1971	Computer Science
M.S.	Cornell University	1964	Engineering Physics
B.S.	University of Wisconsin	1959	Electrical Engineering

Employment

Employer	*Years*	*Position*	*Discipline*
Portland State Univerisity	2005-	Professor Emeritus	Computer Science
National Univ. of Ireland, Galway	2003-04	E.T.S. Walton Fellow	Mathematics
National Univ. of Ireland, Galway	1998-99	Fulbright Scholar	Mathematics
Portland State University	1997-98	Chairman	Computer Science
University College Galway	1996	Visitor	Mathematics
Portland State University	1988-2005	Professor	Computer Science
Oregon Graduate Center	1984-88	Professor	Computer Science
University of Melbourne	1982	Visiting Lecturer	Computer Science
IBM - FSD	1978-83	Consultant	
University of Maryland	1977-84	Associate Professor	Computer Science
Naval Research Laboratory	1976-78	Consultant	
University of Maryland	1971-77	Assistant Professor	Computer Science
Systems Computing, Inc.	1969-70	System Programmer	
Computer Center Corporation	1968-69	Programming Director	
University of Washington	1966-68	Systems Supervisor	
University of Washington	1964-66	Research Assistant	
Shimer College	1962-64	Intern	Natural Sciences
Cornell University	1960-62	Teaching Assistant	Physics
Cornell University	1960-61	Research Assistant	Engineering Physics
University of Wisconsin	1958-59	Technician	Meteorology

Refereed Publications

(All publications are sole-author except as noted. With very few exceptions, the author order is alphabetical.)

Dissertation

Partial recursive computation, 1971, directed by Robert W. Ritchie.

Books

1. *The Engineering of Software* (with Joe Maybee), Addison-Wesley, 2001, 494pp.
2. *Principles of Computer Programming: A Mathematical Approach* (with H. Mills, J. Gannon, V. Basili), Allyn and Bacon, 1987, 669pp.
3. *Introduction to Computation Theory*, Intext Educational Publishers, New York, 1974, 197pp.

Book Chapters

1. Properties of software systems synthesized from components (with Dave Mason and Denise Woit), Chapter 6 of *Component-based Software Development: Case Studies*, K-K. Lau, Ed., World Scientific, 2004.
2. Random testing (updated article), and Subdomain testing, in *Encyclopedia of Software Engineering, 2nd ed.*, J. Marciniak, ed., Wiley, 2005.
3. Editors introduction, software testing and quality assurance, *Annals of Software Engineering* v. 4, Baltzer Science Publishers, 1997.
4. Software quality, software process, and software testing, *Advances in Computers*, M. Zelkowitz, ed., 1995, 1402-1411.
5. Random testing, in *Encyclopedia of Software Engineering*, J. Marciniak, ed., Wiley, 1994, 970-978.
6. Testing programs to detect malicious faults, in *Dependable Computing for Critical Applications 2*, J. Meyer and R. Schlichting, eds., Springer-Verlag, 1992, 375-392.
7. Testing for trustworthiness, in *Directions and Implications of Advanced Computing*, J. P Jacky and D. Schuler, eds., Ablex Publishing Corp, 1989, 97-104.
8. A disciplined text environment, in *Text Processing and Document Manipulation*, J. C. van Vliet, ed., Cambridge University Press, 1986, 78-89.
9. Functional semantics of modules (with J. Gannon & H. Mills), in *Lecture Notes in Computer Science* 186, H. Ehrig, C. Floyd, M. Nivat, and J. Thatcher, eds., Springer-Verlag, 1985, 42-59.

Articles

1. Software component composition: a subdomain-based testing-theory foundation, *J. Software Testing, Verification and Reliability* (June, 2007).
2. Axiomatically checking an implementation against its formal specification (with S. Antoy), *IEEE Trans. Software Engineering SE-26* (January, 2000), 55-69.
3. Evaluating testing methods by delivered reliability, (with P. Frankl, B. Littlewood, and L. Strigini), *IEEE Trans. Software Engineering* SE-24 (Aug., 1998), 586-601.
   Correction, *IEEE Trans. Software Engineering* SE-25 (Mar., 1999), 286.

4. Implementing prototype testing tools, *Software--Practice & Experience*, April, 1995, 347-372.
5. Are we testing for true reliability?, *IEEE Software*, July, 1992, 21-27.
6. Partition testing does not inspire confidence, (with R. Taylor), *IEEE Trans. Software Engineering* SE-16 (Dec., 1990), 1402-1411.
7. New answers to old questions, ("QualityTime" column edited by V. Shen), *IEEE Software*, September, 1990, 89-90,92.
8. Editor's introduction, Special Section on Software Testing, *CACM* 31 (June, 1988), 662-667.
9. A first course in computer science: mathematical principles for software engineering (with H. Mills, J. Gannon, V. Basili), *IEEE Trans. Software Engineering* SE-15 (May, 1989), 550-559.
10. Theory of modules (with J. Gannon & H. Mills), *IEEE Trans. Software Engineering* SE-13 (July, 1987), 820-829.
11. Probable correctness theory, *Info. Proc. Letters.* 25 (April, 1987), 17-25.
12. Software engineering practices in the US and Japan, *Computer* 17 (June, 1984), 57-66 (with M. Zelkowitz, et al.).
13. Hard-to-use criteria for software engineering, *J. Sys. & Software Sci.*, 2 (1981), 89-96.
14. Data abstraction implementation, specification, and testing, *TOPLAS* 3 (July, 1981), 211-223 (with J. Gannon & P. McMullin).
15. Reliability theory of program testing, *Acta Informatica* 16 (1981), 31-43.
16. Transportable package software, *Software - Practice & Experience* 10 (Dec., 1980), 1009-1027 (with R. M. Haralick).
17. Syntax and Semantics of universal programming languages, *Int. J. of Computer Math.* 6 (1977), 87-103.
18. Execution traces and programming-language semantics, *Int. J. of Comp. & Sys. Sci.* 6 (Dec., 1977), 263-279.
19. Testing programs with the aid of a compiler, *IEEE Trans. on Software Eng.* SE-3 (July, 1977), 279-290.
20. Testing programs with finite sets of data, *The Computer J.* 20 (Aug., 1977), 232-237.
21. High-level binding with low-level linkers, *CACM* 19 (Nov., 1976), 642-644.
22. User-like executives, *Software - Practice & Experience* 4 (Jan.-Mar., 1974), 41-50.
23. Efficient multiprogramming resource allocation and accounting *CACM* 16, (June, 1973), 337-343.

Conference Proceedings

1. Test-based specifications of components and systems, First International Workshop on Software Testing and Analysis, Portland, OR, October, 2007, in Proceedings QSIC 2007, 388-395.
2. When only random testing will do, Proceedings First International Workshop on Random Testing, Portland, ME, July, 2006.
3. Subdomain Testing of Units and Systems with State, Proceedings International Symposium on Software Testing and Analysis (ISSTA), Portland, ME, July, 2006, 85-96.
4. Defining 'predictable assembly', Proceedings 9th Symposium on Component-based Software Engineering (CBSE), Vasteras, Sweden, June, 2006, 320-327.

5. Invariants and state in testing and formal methods, Proceedings Program Analysis for Software Tools and Engineering (PASTE), Lisbon, September, 2005, 48-51.
6. On formal specification of software components and systems (with Sharon Flynn), Third Irish Conference on Mathematical Foundations of Computer Science and Information Technology, Dublin, July, 2004. ENTCS 161, 91-107, August, 2006.
7. Experiments with composing component properties, 6th Workshop on CBSE, ICSE 2003, Portland, OR.
8. Continuity in software systems, Proceedings International Symposium on Software Testing and Analysis (ISSTA), Rome, July, 2002, 196-200.
9. Software components: the problem of scale, Workshop Proceedings, ICSE 2001, 4th Workshop on CBSE, Toronto, Canada, June, 2001, 75-80.
10. Theory of software reliability based on components (with D. Mason and D. Woit), International Conference on Sofware Engineering (ICSE), Toronto, Canada, 361-370, May, 2001.
11. On subdomains: testing, profiles, and components, *Proc. ISSTA '00,* Portland, OR, June, 2000, 71-76.
12. Theory of system reliability based on components (with D. Mason and D. Woit), *Workshop Proceedings, ICSE 2000, 3rd Workshop on CBSE,* Limerick, Ireland, May, 2000.
13. Foundational theory of software component reliability, *Proc. FastAbstracts, ISSRE '99,* Boca Ratan, FL, Nov., 1999, 35-36.
14. Keeping the "engineering" in software engineering, *Proc. Quality week '97,* San Francisco, May, 1997.
15. Choosing a testing method to deliver reliability, *Proc. ICSE 19,* Boston, May, 1997, 68-78 (with P. Frankl, B. Littlewood, and L. Strigini).
16. Predicting dependability by testing, *Proc. ISSTA '96,* San Diego, January, 1996, 84-91.
17. Foundations of software testing: dependability theory, *Proc. 2nd Symposium on the Foundations of Software Engineering,* New Orleans, LA,, Dec., 1994, 128-139.
18. Connecting test coverage to software dependability, *Proc. 5th International Symposium on Software Reliability Engineering,* Monterey, CA,, Nov., 1994, 158-165.
19. Exploring dataflow testing of arrays, (with B. Gifford and B. Nikolik), *Proc. 15th ICSE,* Baltimore, May, 1993, 118-129.
20. Faults on its sleeve: amplifying software reliability testing, (with J. Voas), *Proc. ISSTA '93,* Cambridge, June, 1993, 89-98.
21. Testability as ease of establishing reliability, *Proc. Symposium on Issues in Software Reliability Estimation,* Livingston, NJ, October, 1992, 39-52.
22. Self-checking against formal specifications, (with S. Antoy), *Proc. Int. Conf. on Computing and Information*, Toronto, May, 1992, 355-360.
23. Exploring Dataflow Testing with a Novel Analyzer, *Int. Conf. in* Software Engineering Abstracts, Melbourne, Australia, May, 1992, 2-4.
24. Self-checking objects, (with S. Antoy), *Proc. Irvine Software Symposium,* Irvine, CA, March, 1992, 29-48.
25. Comparison of Program Testing Strategies, (with E. Weyuker and S. Weiss), *Proc. TAV-4,* October, 1991, Victoria, BC, 1-10.
26. Testing Programs to Detect Malicious Faults, *Proc. Dependable Computing for Critical Applications,* Tucson, AZ, February, 1991, 162-169.

27. Theoretical comparison of testing methods, *Proc. TAV-3,* December, 1989, Key West, FL.

28. Unit testing for software assurance, *Proc. COMPASS 89,* Washington, DC, 1989, 42-48.

29. Partition testing does not inspire confidence, Proceedings Workshop on Software Testing, Banff, Alberta, July, 1988, 206-215 (with R. Taylor).

30. A first course in computer science: mathematical principles for software engineering, *Proc. SEI Conference on Software Engineering Education,* Springer-Verlag, 1987 (with H. Mills, J. Gannon, V. Basili).

31. Teaching principles of computer programming, *Proc. ACM 15th Annual Computer Science Conference,* St. Louis, 1987 (with H. Mills, J. Gannon, V. Basili).

32. Proposal for a software design control system (DCS), *Proceedings 4th Pacific Northwest Software Quality Conference,* Portland, OR, November, 1986, 265-272 (with R. Babb).

33. Testing for probable correctness, *Proceedings Workshop on Software Testing,* Banff, Alberta, July, 1986, 92-97.

34. Specification theory, *Proceedings 3rd International Workshop on Software Specification and Design,* London, August, 1985, 91-93.

35. Functional semantics of modules, RELCOMEX '84, Ksiaz Castle, Poland, May, 1984, 321-328 (with J. Gannon).

36. Critique of software measurement, *The Measurement of Computer Software Performance*, Los Alamos, August, 1983.

37. Step-wise debugging, *ACM Symposium on High-level Debugging*, Pacific Grove, CA, March, 1983, 198-201.

38. "Determining" tests, *Workshop on effectiveness of testing and proving methods*, Avalon, CA, May, 1982, 87-93.

39. Program maintenance--a modest theory, *Proc. Hawaii International Conference on System Sciences*, Honolulu, January, 1982, 21-26.

40. Hard-to-use evaluation criteria for software engineering, *First ACM SIGSOFT Software Engineering Symposium*, June, 1981.

41. Critique of reliability theory, *Digest of papers*, IEEE Workshop on Software Testing, Ft. Lauderdale, Fla., Dec., 1978, 56-96.

42. Test reliability and software maintenance, *Proc. COMPSAC 78*, Chicago, November, 1978, 315-320.

43. Compile-time testing, *6th Texas conference on Computer Systems*, Austin, November, 1977, 1A15-21.

44. Single-language small-processor systems, *Information Processing 77*, IFIP Congress August, 1977, 969-974.

45. Minicomputer software development: a radical proposal, *Proc. IEEE Trends and Applications*, Gaithersburg, 1976, 107-112.

46. SIMPL systems programming on a minicomputer, *Digest of Papers*, Ninth Annual IEEE Computer Society International Conference, Washington, 1974, 203-206 (with M.V. Zelkowitz).

47. A patent problem for abstract programming languages: machine-independent computations, *Proceedings 4th ACM Symposium on Theory of Computing*, Denver, 1972, 193-197.

Completed Works

1. Functional analysis of programs (with H. Mills), submitted to *Computing Surveys,* Nov., 1984.
2. *What Can Programs Do?,* under contract to Van Nostrand-Reinhold, 1981, 250pp.

Non-refereed Publications

(All publications are sole-author except as noted. With very few exceptions, the author order is alphabetical.)

Books

1. *Structured Computability,* Lecture Note LN-6, Department of Computer Science, University of Maryland, College Park, 1978.
2. *SIMPL-XI: An Introduction to High-Level Systems Programming*, Lecture Note LN-4, Department of Computer Science, University of Maryland, College Park, 1976.
3. *Introduction to Theory of Computation*, Computer Science Center, University of Maryland, College Park, 1972 (Lecture Note LN-3.) Revised edition, 1973 (Lecture Note LN-3'.)
4. *The Electromagnetic Theory,* Shimer College, 1963.

Newsletter Articles

1. Unstructured Gödel numbers, SIGPLAN *Notices* 15 (June, 1980), 8-9.
2. A further note on symmetric keyword pairs, SIGPLAN *Notices* 15 (June, 1980), 7.
3. Testing traversable stacks, SIGPLAN *Notices* 5 (Jan., 1980), 58-65, (with J. Gannon et al.).
4. Report on Florida Testing conference, *Software Engineering Notes* 4 (April, 1979), 17-18.
5. Ignorance of ALGOL 68 considered harmful, SIGPLAN *Notices* 12 (April, 1977), 51-56: correction *ibid* (September, 1977), 17-20.
6. Application of dovetailing to program testing, SIGACT *News* 8 (April, 1976), 25-26.
7. Using the PDP-11 as B5500 for teaching systems programming, SIGPLAN *Notices* 11 (May, 1976), 47-52.
8. Other people's monitors, SIGPLAN *Notices* 8 (July, 1973), 21-22.

Reviews

(More than 40 short reviews of computer science articles appeared in *Computing Reviews*, 1971-1982.)

1. Review of Kfoury, Moll, and Arbib, *A Programming Approach to Computability*, Springer-Verlag, 1982, *Math. Reviews*, 1984.
2. Review of Chen, "On the relationship between computed functions and fixpoints of nondeterministic recursive definitions," *Math. Reviews*, 1983.
3. Review of Glushkov, "Incompleteness theorem of formal theories from programmer's viewpoint," *Math. Reviews*, 1980.
4. Review of Machtey, et al., "Simple Gödel numberings, isomorphisms, and programming properties," *Math. Reviews*, 1978.

Technical Reports (not otherwise published)

1. Computer-assisted writing, Portland State University, Portland, 1989 (TR 89-10).
2. Testing programs to detect sabotage, Portland State University, Portland, 1989 (TR 89-8).
3. Unit testing for software assurance, Portland State University, Portland, 1989 (TR 89-7).
4. Release testing for probable correctness, Oregon Graduate Center, Beaverton, 1984 (TR CS/E 85-003).
5. Functional analysis of programs, Oregon Graduate Center, Beaverton, 1984 (TR CS/E 84-006) (with H. Mills).
6. Debug testing and confidence testing, Oregon Graduate Center, Beaverton, 1984 (TR CS/E 84-004).
7. The software industry: a state of the art survey, Computer Science, University of Maryland, College Park, 1983 (TR-1290) (with M. Zelkowitz et al.).
8. Functional semantics, Computer Science, University of Maryland, College Park, 1983 (TR-1238) (with H. Mills).
9. Step-wise debugging, Department of Computer Science, University of Melbourne, Parkville, 1982 (TR 82/16).
10. Three approaches to program testing theory, Department of Computer Science, University of Melbourne, Parkville, 1982 (TR 82/15).
11. Survey of program testing theory, Department of Computer Science, University of Melbourne, Parkville, 1982 (TR 82/14).
12. Testing of concurrent programs and partial specifications, Department of Computer Science, University of Melbourne, Parkville, 1982 (TR 82/13). (Also position paper for a panel session at Hawaii International Conference on System Sciences, Honolulu, January, 1983.)
13. Theoretical issues in software engineering, Department of Computer Science, University of Melbourne, Parkville, 1982 (TR 82/8).
14. Error propagation and elimination in computer programs, Computer Science, University of Maryland, College Park, 1981 (TR-1065) (with L. Morell).
15. The structure of specifications and implementations of data abstractions, Computer Science, University of Maryland, College Park, 1979 (TR-801) (with M. Ardis).
16. Transportable "package" software, Computer Science, University of Maryland, College Park, 1978 (TR-706) (with R.M. Harlick).
17. Compiler-based systematic testing, Computer Science, University of Maryland, College Park, 1975 (TR-423).
18. On execution traces with an application to the problem of understanding programs, Computer Science, University of Maryland, College Park, 1975 (TR-421).
19. Support of small computers by large, Computer Science, University of Maryland, College Park, 1975 (TR-368).
20. Syntax and semantics of abstract programming languages, Computer Science, University of Maryland, College Park, 1975 (TR-367).
21. Friedberg programming languages. Computer Science Center, University of Maryland, College Park, 1974 (TR-337).
22. Universal abstract programming languages. Computer Science Center, University of Maryland, College Park, 1974 (TR-295).

23. On descriptors and normal state, a note on the Burroughs B5500, Computer System Research Project Technical Note 73-21, University of Maryland Computer Science Center, College park, 1972.
24. On programs and partial recursive functions, University of Washington, Computer Science Group, Seattle, 1970 (Technical Report 70-09-02).

Invited Talks

1. "Testing-based Theory of Predictable Assembly", 2nd workshop on predictable software component assembly, Manchester, U.K., September, 2005.
2. "Lessons about Testing and Formal Specification from Software Component Theory", keynote talk, Microsoft/University of Washington Summer Institute on Testing, Skamania, WA, August, 2004.
3. "Software Component Synthesis Theory: a Subdomain-testing Approach", workshop on predictable software assembly, University of Manchester, U.K., May, 2004. (Also delivered at Centre for Software Reliability, City University, London, and CRN, Pisa, Italy.)
4. "Science, Computer ‘Science’, Mathematics, and Software Development", keynote address, Quality Week ‘02, San Francisco, September, 2002. (Won the ‘best talk’ award.)
5. "Checking Formal Specifications by Testing", keynote address, Irish Workshop on Formal Methods, National University of Ireland, Galway, July, 1999.
6. "Mathematics, Science, Software Engineering," keynote address, First Irish Workshop on Algebra and Topology in Computer Science, Cork, July, 2000.
7. "Software and Society," Fulbright Alumni lecture, National University of Ireland, Galway, March, 1999.
8. "Testing to Deliver Software Reliability", Department of Mathematics, National University of Ireland, Galway, October, 1998.
9. "Keeping the ‘Engineering’ in Software Engineering," keynote address, Quality Week ‘97, San Francisco, May, 1997.
   Preliminary version, keynote address, Pacific Northwest Software Quality Conference, Portland, OR, October, 1996.
10. "Software Quality, Process, Testing," Carnegie-Mellon University, Pittsburgh, PA, February, 1995.
11. "Survey of current research in testing for quality," Pacific Northwest Software Quality Conference, Portland, OR, October, 1994.
12. "Amplifying software reliability," Queens University, Kingston, Ont., January, 1994.
13. "How and why to build prototype testing tools," McMaster University, Hamilton, Ont., January, 1994; AT&T Bell Laboratories, Murray Hill, NJ, October, 1992.
14. "Nobody loves reliability except me and thee, and I’m not too sure about me," (after-dinner speech) Int. Symposium on Software Reliability, Denver, CO, November, 1993.
15. "Why I don’t trust reliability," (Panel), Workshop on software reliability, Boulder, CO, November, 1993.
16. "Not much software reliability from software testing," (Panel), Int. Symposium on Software Reliability, Raleigh, NC, October, 1992.
17. "Theory of software testing and software reliability," National Institute of Science and Technology, Gaithersburg, MD, October, 1992.

18. "Software testing for software reliability," Int. Conf. on Software Engineering, Melbourne, May, 1992.
19. "Experiments with prototype testing tools," University of Maryland, College Park, October, 1991.
20. "Self-checking Objects," Mentor Graphics Corp., Wilsonville, OR, August, 1991
21. "Can tested software be trusted?", Software Research Quality Week, San Francisco, CA, May, 1991
22. "Software reliability and testing," keynote address, Software Reliability Symposium, Denver, CO, May, 1991.
23. "How and why to build prototype testing tools," Purdue University, W. Lafayette, IN, October, 1990.
24. "Survey of program testing with an application to detecting sabotage," University of Victoria, Victoria, BC, Canada, October, 1989.
25. "Testing techniques for quality assurance," Aston-Tate, Inc., Walnut Grove, CA, September, 1989.
26. "An overview of software testing," OCATE workshop on realtime testing, Beaverton, OR, September, 1989.
27. "Foundations of program testing; can tested software be trusted?" Software Research Quality Week, San Francisco, CA, May, 1989; Tektronix Computer Research Laboratory, Beaverton, OR, February, 1989.
28. "What works in software testing," Aston-Tate, Inc., Los Angeles, CA, July, 1989.
29. "Software testing: theory and practice," OCATE workshop on realtime testing, Beaverton, OR, June, 1989.
30. "Testing software for quality," 1989 Northwest quality and reliability conference, Portland, OR, April, 1989.
31. "Theory of modules," Portland State University, Portland, OR, June, 1988.
32. "A proposal for computer-assisted writing," Oregon State University, Corvallis, OR, February, 1987.
33. "How not to evaluate software," Seattle University, Seattle, WA, February, 1987.
34. "Software evaluation," Intel Professional Development Seminar, Portland, OR, March, 1986.
35. "Adversaries in software development," keynote address, Pacific Northwest Software Quality Conference, Portland, OR, September, 1985.
36. "Functional semantics of modules," Technical University of Kielce, Kielce, Poland, May, 1984; University of Arizona, Tucson, AR, October, 1984.
37. "What is a program?," University of North Carolina at Charlotte, Charlotte, NC, March, 1984.
38. "Software engineering: technical discipline or management technique?" Oregon Graduate Center, Portland, OR, November, 1983; Seattle University, Seattle, WA, February, 1984.
39. "Software engineering: I. Specification; II. Testing; III. Maintenance," University of Wollongong, Wollongong, New South Wales, Australia, November, 1982.
40. "Theoretical issues in software engineering; I. The software engineering cycle; II. Specification; III. Testing," Winter School in Theoretical Computer Science, Brisbane, Queensland, Australia, July 1982.

41. "Testing Theory," University of Illinois, Urbana, IL, April, 1982.
42. "The technical side of testing," Wang Institute, Tyngsboro, MA, March, 1982.
43. "Program maintenance," Wang Institute, Tyngsboro, MA, March, 1982.
44. "Testing vs. proving; machines and people," Courant Institute, New York University, New York, N.Y., March 1982; University of Sydney, Sydney, New South Wales, Australia, September, 1982.
45. "What is a program and why doesn't it work?", St. Olaf College, Northfield, Minnesota, June 1981.
46. "Data abstraction--implementation, specification, and testing," University of Victoria, Victoria, British Columbia, May, 1979; University of Washington, Seattle, Washington, May, 1979; University of Melbourne, Parkville, Victoria, Australia, June, 1982; Monash University, Clayton, Victoria, Australia, July, 1982; University of Queensland, Brisbane, Queensland, Australia, July, 1982; University of Wollongong, Wollongong, New South Wales, November, 1982; University of Adelaide, Adelaide, South Australia, Australia, November 1982.
47. "Potential of program-testing tools," Navy technology-transfer conference, Falls Church, Virginia, April, 1978.
48. "SIMPL experiments with a PDP-11," University of Waterloo, Waterloo, Ontario, January, 1977.
49. "Program testing by compiler," Naval Research laboratory, Washington, D.C., March, 1976.

Grants and Fellowships

(Principal investigator except co-p.i. as noted.)

Source	*Description*	*Year*	*Amount*
Science Foundation, Ireland	Formal methods	2003-2004	$200,000
National Science Foundation	Software Components	2001-2005	$300,000
REU Supplement		2002	$7,500
REU Supplement		2003	$6,500
EPSRC (U.K.)	Visiting fellowship	1999	Stg9,000
Fulbright research scholarship	Software engineering	1998-1999	$30,000
Texas Instruments	Institutional	1996	$12,000
Oregon Reg. Strategies Board (with W. Harrison)	Testing Laboratory	1994-1995	$134,958
National Science Foundation	Testing theory	1991-1993	$180,539
Tektronix Foundation (with W. Harrison, L. Shapiro)	Curriculum development	1989-1992	$360,000
National Science Foundation	Software testing	1988-1990	$97,544
REU Supplement		1989	$8,300
RUI Supplement		1989	$20,800
Air Force Office of Scientific Research	Logic programming	1986-1987	$47,000
Allyn & Bacon	Book preparation	1985	$7,000
Australian Government	Overseas scholar grant	1982	A$5,000
International Business Machines (with R. Yeh et al.)	Software productivity	1982-1983	$125,000
Renewal		1984	$35,000

Air Force Office of Scientific Research (with J. Gannon)	Data abstraction/testing	1979-1980	$52,000
Renewal (with V. Basili et al.)		1980-1982	$220,000
Renewal		1982-1983	$240,000
Renewal		1984-1985	$299,000
National Science Foundation (with A. Rosenfeld)	Transportable image processing	1978-1979	$105,000
Renewal		1980-1981	$75,000
General Research Board		1975	$2,500
Defense Mapping Agency	Compiler development	1974-1975	$25,370
Radio Corporation of America	Fellowship	1960-1961	$2,500

Research in Progress

Program testing theory.

I am investigating the foundations of program testing, particularly a statistical theory of program "dependability." This theory seeks to predict the quality of software independent of its usage, and do so with measurements that are feasible in practice. The present focus of the research is on software components.

Teaching Achievements

Theses directed

Degree	*Student*	*Thesis/paper(P) title*	*Date*
Ph.D.	Borislav Nikolik	Reliability of Programs Specified with Equational Specifications (joint advisor, Zary Segal)	1998
Ph.D.	Clifford Walinsky	Constructive Negation in Logic Programs	1987
Ph.D.	Larry Morell	Theory of Error-based Testing	1983
Ph.D.	Mark Ardis	Data Abstraction Transformations	1980
M.S.	James Campbell	Dataflow Regression Testing	1990
M.S.	Richard Broussard	Prototype for a Document-preparation Environment	1986
M.S.	Thomas Hudson	Stepwise Debugging	1984
M.S.	Albert Lang, Jr.	An Investigation of the General Programming Charts	1978
M.S.	Kevin Casey	UNIVAC 1108 to DEC 1070 SIMPL-X Bootstrap (P)	1976
M.S.	Steve Kaiser	Operating System Command Languages	1975
M.S.	John Barkley	A Central Laboratory Automation Facility (P)	1974
M.S.	Robert Rockwell	Survey of Monitor-Subordinate Processor Systems	1974
M.S.	Helen Carter	Code Optimization of Arithmetic Expressions in Stack Environments	1974
M.S.	Mary Fitzgerald	A Note on ALGOL 68 Syntax and Semantics (P)	1973

Courses taught

(Almost all the courses numbered above 400 were developed—and redeveloped—each time they were taught. In addition, a textbook and complete supporting materials were written for an introductory course (CMSC 122), for an intermediate course (CMSC 452), and for two graduate courses (CMSC 600 and CMSC 640). I also designed a systems-programming language for CMSC 600, and implemented it on the PDP-11.)

Shimer College

Natural Science I - Chemistry
Natural Science II - Biology
Natural Science III - Physics
Natural Science IV - Physics and biology
Humanities II - Introduction to literature
Mathematics I - Logic
Mathematics III - Calculus
Mathematics VI - Advanced calculus
Mathematics VII - Complex variables

University of Maryland

CMSC 103	Introductory algorithmic methods
CMSC 122	Computer Science II
HONR 140	The digital computer
CMSC 210	Language and structures of computers
CMSC 314	Introduction to computer languages and systems
CMSC 330	Programming language constructs
CMSC 415	Systems programming
CMSC 430	Compiler design
CMSC 440	Structure of programming languages
CMSC 450	Elementary logic and algorithms
CMSC 452	Elementary theory of computation
CMSC 455	Formal languages
CMSC 498	Computer executive software
CMSC 600	Programming systems
CMSC 630	Theory of programming languages
CMSC 640	Automata and computability
CMSC 750	Theory of computability
CMSC 798	Seminar in ALGOL 68
CMSC 798	Seminar in programming languages
CMSC 798	Seminar in software engineering
CMSC 818	Practical systems programming
CMSC 818	Seminar in operating systems
CMSC 838	Testing and certification
CMSC 838	Software engineering
CMSC 858	Computational complexity

University of Melbourne

Software testing seminar

Oregon Graduate Center

CS/E 480	Introduction to theoretical computer science
CS/E 500	Introduction to software engineering
CS/E 504	Program verification
CS/E 513	Operating system principles
CS/E 533	Automata and formal languages
CS/E 581	Software testing and verification

Portland State University

CS 554	Software engineering
CS 300	Introduction to software engineering
CS 510ST	Software testing

CS 581 Theory of computing
CS 303 Operating systems
CS 431-32 Operating systems
CS 556 Software lifecycle--implementation and testing
CS 595 Programming languages seminar
CS 510TV Testing and verification
CS 458 Design of programming languages
CS 410IL Structure of programming languages
OMSE 525 Software quality
OMSE 535 Implementation and testing

Membership in Professional Societies

Association for Computing Machinery
Special interest groups SIGACT, SIGPLAN, SIGSOFT, SIGOIS
IEEE Computer Society
Computer Professionals for Social Responsibility

Service to Profession

Reviewer for *CACM, IEEE Software, IEEE Computer, IEEE Trans. on Software Engineering, Math. Rev., TOPLAS, TOSEM, Software--Prac. & Exp., Computer J.* NSF, AFOSR.

Co-chair, Doctoral Symposium, ICSE 2003.

Program Committee Co-chair, COMPASS, 1997.

General Chair, ISSTA, Seattle, 1993.

Program committee: CBSE Workshop, 2002-; FATES, 2002-2004; DCCA, 1996; ISSRE, 1994; ISSTA, 1992, 1996, 2002; PNSQC, 1990-2000; Quality Week, 1994-2000.

Editorial board *IEEE Trans. on Software Engineering* 1994-2000, *J. Systems & Software* 1992-2001, *J. Software Testing, Verification, and Reliability*, *Software Quality J.*

Service to University, College, Department

Departmental

Graduate Committee (Chair) 1999-2003

Promotion and Tenure Committee Chair 1988-89, 1991-6, 1998-.

Colloquium Committee, 1988-89, 2004-2005.

Recruiting Committee, 1989-90, 1992-93, 1997-98.

Established Faculty Seminar, 1989.

Service to Community

Pacific Northwest Software Quality Conference
   Board member, 1987-1991, 1996-1998.
   Vice President, 1987-89
   Program chair, 1987-89, 1992, 1997

Mentor for Apprenticeships in Science and Engineering, 1990-2004.

Judge for Northwest Science Expo (highschool science talent contest), 1986-1992.

Organized OCATE Realtime Workshop, June, 1989 (with Bob Glass).