MICHAEL ERNST

Curriculum Vitae

Computer Science & Engineering
University of Washington
Allen Center 538
Box 352350
Seattle, WA 98195-2350

Phone: 206-221-0965
Fax: 206-616-3804
Email: mernst@uw.edu
Web: https://homes.cs.washington.edu/ mernst/

EDUCATIONAL HISTORY

University of Washington, Seattle
Ph.D., Computer Science & Engineering
August 2000
Dissertation: Dynamically Discovering Likely Program Invariants

University of Washington, Seattle
M.S., Computer Science & Engineering
March 1997

Massachusetts Institute of Technology, Cambridge, MA
S.M., Electrical Engineering and Computer Science
September 1992

Massachusetts Institute of Technology, Cambridge, MA
S.B., Electrical Engineering and Computer Science
June 1989

EMPLOYMENT HISTORY

University of Washington
Seattle, WA
Professor, Sep. 2014 – present
Associate Professor, Jan. 2009 – Sep. 2014

MIT
Cambridge, MA
Associate Professor, July 2007 – Dec. 2008
Associate Professor (without tenure), Feb. 2005 – June 2007
Assistant Professor, Sep. 2000 – Feb. 2005

University of Washington
Seattle, WA
Research Assistant, Sept. 1996 – Aug. 2000

Rice University
Houston, TX
Lecturer, Sept. 1995 – May 1996

Microsoft
Redmond, WA
Researcher, Mar. 1993 – Aug. 1995
Software Design Engineer, Sep. 1992 – Mar. 1993

MIT
Cambridge, MA
Research Assistant, Sep. 1989 – Aug. 1992
Teaching Assistant, Jan. 1989 – June 1989

Texas Instruments
Dallas, TX
Summer intern, summers May 1986 – Sep. 1989

Technical consulting:
Facebook, June 2018 – September 2018
Samsung, Dec. 2009 – Dec. 2010
McKinsey & Co., Sep. 2007 – Jan. 2008
Kestrel Technology, Nov. 2006 – Dec. 2008
Institute for Defense Analyses, Jan. 2006 – Dec. 2008
Mercury Interactive, July 2004 – July 2005

Legal consulting:
Kelley Goldfarb Huck Rath & Riojas PLLC, November 2017 – November 2018
Sullivan Law Group, January 2016 – June 2016
VendNovation, August 2011 – July 2012
Sugarman & Sugarman, May 2008 – July 2009
GraniteStream, Oct. 2001 – May 2003

AWARDS AND HONORS

Susan Eggers Endowed Professorship in Computer Science & Engineering, 2022
ECOOP Distinguished Artifact Award, 2022
IEEE Fellow, 2021
ACM SIGSOFT Outstanding Research Award, 2020
ISSTA Impact Award, 2019
ACM SIGSOFT Distinguished Paper Award, 2018
CRA-E Undergraduate Research Faculty Mentoring Award, 2018
ISSTA Impact Award, 2018
JavaOne 2017 Rock Star, September 2017
ICSE 2017 Most Influential Paper award, for ICSE 2007 paper, May 2017
Ranked 3rd among all software engineering researchers, AMiner, 2016
JavaOne 2016 Rock Star, September 2016
ASE 2015 paper nominated for Distinguished Paper Award, November 2015
Fellow of the ACM, 2014
FSE 2014 ACM Distinguished Paper Award, November 2014
ISSTA 2014 ACM Distinguished Paper Award, July 2014
Ranked 2nd among software engineering researchers worldwide, for work in the last 10 years, Microsoft

Academic Search, 2013
UIST Best Paper honorable mention, 2013
ACM SIGSOFT Impact Paper Award, 2013
ESEC/FSE 2011 ACM Distinguished Paper Award, September 2011
ECOOP 2011 Best Paper Award, July 2011
Inaugural IBM John Backus Award, August 2009
ISSTA 2009 ACM Distinguished Paper Award, July 2009
JavaOne 2009 Rock Star, May 2009
ISSTA 2008 paper selected for expedited journal publication, July 2008
ASE 2007 paper selected for expedited journal publication, November 2007
ESEC/FSE 2007 ACM Distinguished Paper Award, September 2007
Ranked 5th among software engineering researchers worldwide in a CACM article, June 2007
ICSE 2007 ACM Distinguished Paper Award, May 2007
Most Innovative JSR of the Year (Sun Microsystems JCP Program), May 2007
ASE 2006 paper selected for expedited journal publication, September 2006
ISSTA 2006 paper selected for expedited journal publication, July 2006
IBM faculty award, June 2006
ASE 2005 paper nominated for Best Paper Award, November 2005
ICSE 2004 paper nominated for ACM Distinguished Paper Award, May 2004
Ross Career Development Professorship of Software Technology (MIT), July 2003
ESEC/FSE 2003 ACM Distinguished Paper Award, September 2003
IBM Eclipse Innovation Award 2003, 2004, 2005
VMCAI 2003 paper selected for expedited journal publication, January 2003
NSF CAREER Award, February 2002
ICSM 2001 Best dissertation of past three years, November 2001
Honorable mention, ACM doctoral dissertation competition, 2000
U. of Washington William Chan Memorial Dissertation Award, 2000
ICSE 2000 paper selected for expedited journal publication, June 2000
ICSE 1999 paper selected for expedited journal publication, May 1999

(The list does not include graduate fellowships or other graduate school and earlier awards.)

AFFILIATIONS AND OTHER APPOINTMENTS

Adjunct Associate Professor, Electrical Engineering and Computer Science, MIT, Cambridge, MA, USA

PUBLICATIONS

This section avoids duplications by omitting previous conference, workshop, or TR publications that are superseded by a subsequent publication. It also omits tool demos and other ancillary publications, even when fully reviewed. A full list, with indications of relationships among papers, appears at `http://homes.cs.washington.edu/~mernst/pubs/`. Also note that in computer science, conference articles are at least as prestigious as journal articles.

Refereed archival journal publications

[1] Colin S. Gordon, Michael D. Ernst, Dan Grossman, and Matthew Parkinson. Verifying invariants of lock-free data structures with rely-guarantee and refinement types. *ACM Transactions on Programming Languages and Systems*, 39(3):11:1–11:54, May 2017.

[2] Ivan Beschastnikh, Patty Wang, Yuriy Brun, and Michael D. Ernst. Debugging distributed systems: Challenges and options for validation and debugging. *Communications of the ACM*, 59(8):32–37, August 2016.

[3] Ivan Beschastnikh, Patty Wang, Yuriy Brun, and Michael D. Ernst. Debugging distributed systems: Challenges and options for validation and debugging. *ACM Queue*, 14(2):91–110, March/April 2016.

[4] Kıvanç Muşlu, Yuriy Brun, Michael D. Ernst, and David Notkin. Reducing feedback delay of software development tools via continuous analysis. *IEEE Transactions on Software Engineering*, 41(8):745–763, August 2015.

[5] Ivan Beschastnikh, Yuriy Brun, Jenny Abrahamson, Michael D. Ernst, and Arvind Krishnamurthy. Using declarative specification to improve the understanding, extensibility, and comparison of model-inference algorithms. *IEEE Transactions on Software Engineering*, 41(4):408–428, April 2015.

[6] Yuriy Brun, Reid Holmes, Michael D. Ernst, and David Notkin. Early detection of collaboration conflicts and risks. *IEEE Transactions on Software Engineering*, 39(10):1358–1375, October 2013.

[7] Adam Kieżun, Vijay Ganesh, Shay Artzi, Philip J. Guo, Pieter Hooimeijer, and Michael D. Ernst. HAMPI: A solver for word equations over strings, regular expressions, and context-free grammars. *ACM Transactions on Software Engineering and Methodology*, 21(4):25:1–25:28, November 2012.

[8] Yingyi Bu, Bill Howe, Magdalena Balazinska, and Michael D. Ernst. The HaLoop approach to large-scale iterative data analysis. *The VLDB Journal*, 21(2):169–190, 2012.

[9] Ivan Beschastnikh, Yuriy Brun, Michael D. Ernst, Arvind Krishnamurthy, and Thomas E. Anderson. Mining temporal invariants from partially ordered logs. *SIGOPS Operating Systems Review*, 45(3):39–46, December 2011.

[10] Frank Tip, Robert M. Fuhrer, Adam Kieżun, Michael D. Ernst, Ittai Balaban, and Bjorn De Sutter. Refactoring using type constraints. *ACM Transactions on Programming Languages and Systems*, 33(3):9:1–9:47, May 2011.

[11] Shay Artzi, Adam Kieżun, Julian Dolby, Frank Tip, Danny Dig, Amit Paradkar, and Michael D. Ernst. Finding bugs in web applications using dynamic test generation and explicit state model checking. *IEEE Transactions on Software Engineering*, 36(4):474–494, July/August 2010.

[12] Shay Artzi, Jaime Quinonez, Adam Kieżun, and Michael D. Ernst. Parameter reference immutability: Formal definition, inference tool, and comparison. *Automated Software Engineering*, 16(1):145–192, March 2009.

[13] Michael D. Ernst, Jeff H. Perkins, Philip J. Guo, Stephen McCamant, Carlos Pacheco, Matthew S. Tschantz, and Chen Xiao. The Daikon system for dynamic detection of likely invariants. *Science of Computer Programming*, 69(1–3):35–45, December 2007.

[14] Lilian Burdy, Yoonsik Cheon, David Cok, Michael D. Ernst, Joe Kiniry, Gary T. Leavens, K. Rustan M. Leino, and Erik Poll. An overview of JML tools and applications. *Software Tools for Technology Transfer*, 7(3):212–232, June 2005.

[15] Toh Ne Win, Michael D. Ernst, Stephen J. Garland, Dilsun Kırlı, and Nancy Lynch. Using simulated execution in verifying distributed algorithms. *Software Tools for Technology Transfer*, 6(1):67–76, July 2004.

[16] Michael D. Ernst, Greg J. Badros, and David Notkin. An empirical analysis of C preprocessor use. *IEEE Transactions on Software Engineering*, 28(12):1146–1170, December 2002.

[17] Elizabeth L. Wilmer and Michael D. Ernst. Graphs induced by Gray codes. *Discrete Mathematics*, 257:585–598, November 28, 2002.

[18] Michael D. Ernst, Jake Cockrell, William G. Griswold, and David Notkin. Dynamically discovering likely program invariants to support program evolution. *IEEE Transactions on Software Engineering*, 27(2):99–123, February 2001.

[19] Michael D. Ernst. Playing Konane mathematically: A combinatorial game-theoretic analysis. *UMAP Journal*, 16(2):95–121, Spring 1995.

Conference proceedings and other non-journal articles – Fully refereed publications

[20] Martin Kellogg, Narges Shadab, Manu Sridharan, and Michael D. Ernst. Accumulation analysis. In *ECOOP 2022 — Object-Oriented Programming, 33rd European Conference*, pages 10:1–10:31, Berlin, Germany, June 2022.

[21] Zhen Zhang, Yu Feng, Michael D. Ernst, Sebastian Porst, and Isil Dillig. Checking conformance of applications against GUI policies. In *ESEC/FSE 2021: The ACM 29th joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pages 95–106, Athens, Greece, August 2021.

[22] Martin Kellogg, Narges Shadab, Manu Sridharan, and Michael D. Ernst. Lightweight and modular resource leak verification. In *ESEC/FSE 2021: The ACM 29th joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pages 181–192, Athens, Greece, August 2021.

[23] Rashmi Mudduluru, Jason Waataja, Suzanne Millstein, and Michael D. Ernst. Verifying determinism in sequential programs. In *ICSE 2021, Proceedings of the 43rd International Conference on Software Engineering*, pages 37–49, Madrid, Spain, May 2021.

[24] Martin Kellogg, Martin Schäf, Serdar Tasiran, and Michael D. Ernst. Continuous compliance. In *ASE 2020: Proceedings of the 35th Annual International Conference on Automated Software Engineering*, pages 511–523, Melbourne, Australia, September 2020.

[25] Wing Lam, August Shi, Reed Oei, Sai Zhang, Michael D. Ernst, and Tao Xie. Dependent-test-aware regression testing techniques. In *ISSTA 2020, Proceedings of the 2020 International Symposium on Software Testing and Analysis*, pages 298–311, Los Angeles, CA, USA, July 2020.

[26] Annie Louis, Santanu Kumar Dash, Earl T. Barr, Michael D. Ernst, and Charles Sutton. Where should i comment my code? a dataset and model for predicting locations that need comments. In *ICSE NIER, Proceedings of the 42nd International Conference on Software Engineering, New Ideas and Emerging Results Track*, pages 21–24, Seoul, Korea, May 2020.

[27] Martin Kellogg, Manli Ran, Manu Sridharan, Martin Schäf, and Michael D. Ernst. Verifying object construction. In *ICSE 2020, Proceedings of the 42nd International Conference on Software Engineering*, pages 1447–1458, Seoul, Korea, May 2020.

[28] Ellis Michael, Doug Woos, Thomas Anderson, Michael D. Ernst, and Zachary Tatlock. Teaching rigorous distributed systems with efficient model checking. In *EuroSys*, pages 1–15, Dresden, Germany, March 2019.

[29] Martin Kellogg, Vlastimil Dort, Suzanne Millstein, and Michael D. Ernst. Lightweight verification of array indexing. In *ISSTA 2018, Proceedings of the 2018 International Symposium on Software Testing and Analysis*, pages 3–14, Amsterdam, Netherlands, July 2018.

[30] Arianna Blasi, Alberto Goffi, Konstantin Kuznetsov, Alessandra Gorla, Michael D. Ernst, Mauro Pezzè, and Sergio Delgado Castellanos. Translating code comments to procedure specifications. In *ISSTA 2018, Proceedings of the 2018 International Symposium on Software Testing and Analysis*, pages 242–253, Amsterdam, Netherlands, July 2018.

[31] René Just, Chris Parnin, Ian Drosos, and Michael D. Ernst. Comparing developer-provided to user-provided tests for fault localization and automated program repair. In *ISSTA 2018, Proceedings of the 2018 International Symposium on Software Testing and Analysis*, pages 287–297, Amsterdam, Netherlands, July 2018.

[32] Pavel Panchekha, Adam Geller, Michael D. Ernst, Zachary Tatlock, and Shoaib Kamil. Verifying that web pages have accessible layout. In *PLDI 2018: Proceedings of the ACM SIGPLAN 2016 Conference on Programming Language Design and Implementation*, pages 1–14, Philadelphia, PA, USA, June 2018.

[33] Calvin Loncaric, Michael D. Ernst, and Emina Torlak. Generalized data structure synthesis. In *ICSE 2018, Proceedings of the 40th International Conference on Software Engineering*, pages 958–968, Gothenburg, Sweden, May 2018.

[34] Xi Victoria Lin, Chenglong Wang, Luke Zettlemoyer, and Michael D. Ernst. NL2Bash: A corpus and semantic parser for natural language interface to the Linux operating system. In *LREC: Language Resources and Evaluation Conference*, Miyazaki, Japan, May 2018.

[35] Jonathan Jacky, Stefani Banerian, Michael D. Ernst, Calvin Loncaric, Stuart Pernsteiner, Zachary Tatlock, and Emina Torlak. Automatic formal verification for EPICS. In *ICALEPCS 2017: 16th International Conference on Accelerator and Large Experimental Physics Control Systems*, Barcelona, Spain, October 2017.

[36] Konstantin Weitz, Steven Lyubomirsky, Stefan Heule, Emina Torlak, Michael D. Ernst, and Zachary Tatlock. SpaceSearch: A library for building and verifying solver-aided tools. In *ICFP 2017: Proceedings of the 22nd ACM SIGPLAN International Conference on Functional Programming*, pages 25:1–25:28, Oxford, UK, September 2017.

[37] Spencer Pearson, José Campos, René Just, Gordon Fraser, Rui Abreu, Michael D. Ernst, Deric Pang, and Benjamin Keller. Evaluating and improving fault localization. In *ICSE 2017, Proceedings of the 39th International Conference on Software Engineering*, pages 609–620, Buenos Aires, Argentina, May 2017.

[38] Michael D. Ernst. Natural language is a programming language: Applying natural language processing to software development. In *SNAPL 2017: the 2nd Summit oN Advances in Programming Languages*, pages 4:1–4:14, Asilomar, CA, USA, May 2017.

[39] Konstantin Weitz, Doug Woos, Emina Torlak, Michael D. Ernst, Arvind Krishnamurthy, and Zachary Tatlock. Scalable verification of Border Gateway Protocol configurations with an SMT solver. In *OOPSLA 2016, Object-Oriented Programming Systems, Languages, and Applications*, pages 765–780, Amsterdam, November 2016.

[40] Chandrakana Nandi and Michael D. Ernst. Automatic trigger generation for rule-based smart homes. In *PLAS 2016: ACM SIGPLAN Workshop on Programming Languages and Analysis for Security*, pages 97–102, Vienna, Austria, October 2016.

[41] Konstantin Weitz, Doug Woos, Emina Torlak, Michael D. Ernst, Arvind Krishnamurthy, and Zachary Tatlock. Formal semantics and automated verification for the Border Gateway Protocol. In *NetPL 2016: ACM SIGCOMM Workshop on Networking and Programming Languages (NetPL 2016)*, Florianópolis, Brazil, August 2016.

[42] Stuart Pernsteiner, Calvin Loncaric, Emina Torlak, Zachary Tatlock, Xi Wang, Michael D. Ernst, and Jonathan Jacky. Investigating safety of a radiotherapy machine using system models with pluggable checkers. In *CAV 2016: 28th International Conference on Computer Aided Verification*, pages 23–41, Toronto, Canada, July 2016.

[43] Alberto Goffi, Alessandra Gorla, Michael D. Ernst, and Mauro Pezzè. Automatic generation of oracles for exceptional behaviors. In *ISSTA 2016, Proceedings of the 2016 International Symposium on Software Testing and Analysis*, pages 213–224, Saarbrücken, Genmany, July 2016.

[44] Calvin Loncaric, Emina Torlak, and Michael D. Ernst. Fast synthesis of fast collections. In *PLDI 2016: Proceedings of the ACM SIGPLAN 2016 Conference on Programming Language Design and Implementation*, pages 355–368, Santa Barbara, CA, USA, June 2016.

[45] Michael D. Ernst, Damiano Macedonio, Massimo Merro, and Fausto Spoto. Semantics for locking specifications. In *NFM 2016: 8th NASA Formal Methods Symposium*, pages 355–372, Minneapolis, MN, USA, June 2016.

[46] Michael D. Ernst, Alberto Lovato, Damiano Macedonio, Fausto Spoto, and Javier Thaine. Locking discipline inference and checking. In *ICSE 2016, Proceedings of the 38th International Conference on Software Engineering*, pages 1133–1144, Austin, TX, USA, May 2016.

[47] Doug Woos, James R. Wilcox, Steve Anton, Zachary Tatlock, Michael D. Ernst, and Thomas Anderson. Planning for change in a formal verification of the Raft consensus protocol. In *CPP 2016: 5th ACM SIGPLAN Conference on Certified Programs and Proofs*, pages 154–165, St. Petersburg, FL, USA, January 2016.

[48] Michael D. Ernst, Alberto Lovato, Damiano Macedonio, Ciprian Spiridon, and Fausto Spoto. Boolean formulas for the static identification of injection attacks in Java. In *LPAR 2015: Proceedings of the 20th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, pages 130–145, Suva, Fiji, November 2015.

[49] Kıvanç Muşlu, Luke Swart, Yuriy Brun, and Michael D. Ernst. Development history granularity transformations. In *ASE 2015: Proceedings of the 30th Annual International Conference on Automated Software Engineering*, pages 697–702, Lincoln, NE, USA, November 2015.

[50] Paulo Barros, René Just, Suzanne Millstein, Paul Vines, Werner Dietl, Marcelo d'Amorim, and Michael D. Ernst. Static analysis of implicit control flow: Resolving Java reflection and Android intents. In *ASE 2015: Proceedings of the 30th Annual International Conference on Automated Software Engineering*, pages 669–679, Lincoln, NE, USA, November 2015.

[51] Brian Burg, Andrew J. Ko, and Michael D. Ernst. Explaining visual changes in web interfaces. In *UIST 2015: Proceedings of the 28th ACM Symposium on User Interface Software and Technology*, pages 259–268, Charlotte, NC, USA, November 2015.

[52] Irfan Ul Haq, Juan Caballero, and Michael D. Ernst. Ayudante: Identifying undesired variable interactions. In *WODA 2015: 13th International Workshop on Dynamic Analysis*, pages 8–13, Pittsburgh, PA, USA, October 2015.

[53] Drew Dean, Sean Guarino, Leonard Eusebi, Andrew Keplinger, Tim Pavlik, Ronald Watro, Aaron Cammarata, John Murray, Kelly McLaughlin, John Cheng, and Thomas Maddern. Lessons learned in game development for crowdsourced software formal verification. In *3GSE: USENIX Summit on Gaming, Games, and Gamification in Security Education*, Washington, DC, August 2015.

[54] Sai Zhang and Michael D. Ernst. Proactive detection of inadequate diagnostic messages for software configuration errors. In *ISSTA 2015, Proceedings of the 2015 International Symposium on Software Testing and Analysis*, pages 12–23, Baltimore, MD, USA, July 2015.

[55] James R. Wilcox, Doug Woos, Pavel Panchekha, Zachary Tatlock, Xi Wang, Michael D. Ernst, and Thomas Anderson. Verdi: A framework for implementing and formally verifying distributed systems. In *PLDI 2015: Proceedings of the ACM SIGPLAN 2015 Conference on Programming Language Design and Implementation*, pages 357–368, Portland, OR, USA, June 2015.

[56] Mohsen Vakilian, Amarin Phaosawasdi, Michael D. Ernst, and Ralph E. Johnson. Cascade: A universal programmer-assisted type qualifier inference tool. In *ICSE 2015, Proceedings of the 37th International Conference on Software Engineering*, pages 234–245, Florence, Italy, May 2015.

[57] Michael D. Ernst, Dan Grossman, Jon Jacky, Calvin Loncaric, Stuart Pernsteiner, Zachary Tatlock, Emina Torlak, and Xi Wang. Toward a dependability case language and workflow for a radiation therapy system. In *SNAPL 2015: the Inaugural Summit oN Advances in Programming Languages*, pages 103–112, Asilomar, CA, USA, May 2015.

[58] Ruth Anderson, Michael D. Ernst, Robert Ordóñez, Paul Pham, and Ben Tribelhorn. A data programming CS1 course. In *SIGCSE: Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 150–155, Kansas City, MO, USA, March 2015.

[59] René Just, Darioush Jalali, Laura Inozemtseva, Michael D. Ernst, Reid Holmes, and Gordon Fraser. Are mutants a valid substitute for real faults in software testing? In *FSE 2014: Proceedings of the ACM SIGSOFT 22nd Symposium on the Foundations of Software Engineering*, pages 654–665, Hong Kong, November 2014.

[60] Michael D. Ernst, René Just, Suzanne Millstein, Werner Dietl, Stuart Pernsteiner, Franziska Roesner, Karl Koscher, Paulo Barros, Ravi Bhoraskar, Seungyeop Han, Paul Vines, and Edward X. Wu. Collaborative verification of information flow for a high-assurance app store. In *CCS 2014: Proceedings of the 21st ACM Conference on Computer and Communications Security*, pages 1092–1104, Scottsdale, AZ, USA, November 2014.

[61] René Just, Darioush Jalali, and Michael D. Ernst. Defects4J: A Database of existing faults to enable controlled testing studies for Java programs. In *ISSTA 2014, Proceedings of the 2014 International Symposium on Software Testing and Analysis*, pages 437–440, San Jose, CA, USA, July 2014. Tool demo.

[62] René Just, Michael D. Ernst, and Gordon Fraser. Efficient mutation analysis by propagating and partitioning infected execution states. In *ISSTA 2014, Proceedings of the 2014 International Symposium on Software Testing and Analysis*, pages 315–326, San Jose, CA, USA, July 2014.

[63] Konstantin Weitz, Gene Kim, Siwakorn Srisakaokul, and Michael D. Ernst. A type system for format strings. In *ISSTA 2014, Proceedings of the 2014 International Symposium on Software Testing and Analysis*, pages 127–137, San Jose, CA, USA, July 2014.

[64] Sai Zhang, Darioush Jalali, Jochen Wuttke, Kıvanç Muşlu, Wing Lam, Michael D. Ernst, and David Notkin. Empirically revisiting the test independence assumption. In *ISSTA 2014, Proceedings of the 2014 International Symposium on Software Testing and Analysis*, pages 385–396, San Jose, CA, USA, July 2014.

[65] Ravi Bhoraskar, Dominic Langenegger, Pingyang He, Raymond Cheng, Will Scott, and Michael D. Ernst. User scripting on Android using BladeDroid. In *APSys 2014: 5th Asia-Pacific Workshop on Systems*, pages 9:1–9:7, Beijing, China, June 2014.

[66] Jenny Abrahamson, Ivan Beschastnikh, Yuriy Brun, and Michael D. Ernst. Shedding light on distributed system executions. In *ICSE 2014, Proceedings of the 36th International Conference on Software Engineering*, pages 598–599, Hyderabad, India, June 2014.

[67] Todd W. Schiller, Kellen Donohue, Forrest Coward, and Michael D. Ernst. Case studies and tools for contract specifications. In *ICSE 2014, Proceedings of the 36th International Conference on Software Engineering*, pages 596–607, Hyderabad, India, June 2014.

[68] Sai Zhang and Michael D. Ernst. Which configuration option should I change? In *ICSE 2014, Proceedings of the 36th International Conference on Software Engineering*, pages 152–163, Hyderabad, India, June 2014.

[69] Ivan Beschastnikh, Yuriy Brun, Michael D. Ernst, and Arvind Krishnamurthy. Inferring models of concurrent systems from logs of their behavior with CSight. In *ICSE 2014, Proceedings of the 36th International Conference on Software Engineering*, pages 468–479, Hyderabad, India, June 2014.

[70] Ruth Anderson, Michael D. Ernst, Robert Ordóñez, Paul Pham, and Steven A. Wolfman. Introductory programming meets the real world: Using real problems and data in CS1. In *SIGCSE: Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, pages 465–466, Atlanta, GA, USA, March 2014.

[71] Brian Burg, Richard Bailey, Andrew J. Ko, and Michael D. Ernst. Interactive record/replay for web application debugging. In *UIST 2013: Proceedings of the 26th ACM Symposium on User Interface Software and Technology*, pages 473–484, St. Andrews, UK, October 2013.

[72] Kıvanç Muşlu, Yuriy Brun, Michael D. Ernst, and David Notkin. Making offline analyses continuous. In *ESEC/FSE 2013: The 9th joint meeting of the European Software Engineering Conference (ESEC) and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE)*, pages 323–333, St. Petersburg, Russia, August 2013.

[73] Sai Zhang, Hao Lü, and Michael D. Ernst. Automatically repairing broken workflows for evolving GUI applications. In *ISSTA 2013, Proceedings of the 2013 International Symposium on Software Testing and Analysis*, pages 45–55, Lugano, Switzerland, July 2013.

[74] Colin S. Gordon, Werner Dietl, Michael D. Ernst, and Dan Grossman. JavaUI: Effects for controlling UI object access. In *ECOOP 2013 — Object-Oriented Programming, 27th European Conference*, pages 179–204, Montpellier, France, July 2013.

[75] Colin S. Gordon, Michael D. Ernst, and Dan Grossman. Rely-guarantee references for refinement types over aliased mutable data. In *PLDI 2013: Proceedings of the ACM SIGPLAN 2013 Conference on Programming Language Design and Implementation*, pages 73–84, Seattle, WA, USA, June 2013.

[76] Alex Potanin, Johan Östlund, Yoav Zibin, and Michael D. Ernst. Immutability. In *Aliasing in Object-Oriented Programming*, volume 7850 of *LNCS*, pages 233–269. Springer-Verlag, April 2013.

[77] Ivan Beschastnikh, Yuriy Brun, Jenny Abrahamson, Michael D. Ernst, and Arvind Krishnamurthy. Unifying FSM-inference algorithms through declarative specification. In *ICSE 2013, Proceedings of the 35th International Conference on Software Engineering*, pages 252–261, San Francisco, CA, USA, May 2013.

[78] Sai Zhang and Michael D. Ernst. Automated diagnosis of software configuration errors. In *ICSE 2013, Proceedings of the 35th International Conference on Software Engineering*, pages 312–321, San Francisco, CA, USA, May 2013.

[79] Wei Huang, Ana Milanova, Werner Dietl, and Michael D. Ernst. ReIm & ReImInfer: Checking and inference of reference immutability and method purity. In *OOPSLA 2012, Object-Oriented Programming Systems, Languages, and Applications*, pages 879–896, Tucson, AZ, USA, October 2012.

[80] Todd W. Schiller and Michael D. Ernst. Reducing the barriers to writing verified specifications. In *OOPSLA 2012, Object-Oriented Programming Systems, Languages, and Applications*, pages 95–112, Tucson, AZ, USA, October 2012.

[81] Kıvanç Muşlu, Yuriy Brun, Reid Holmes, Michael D. Ernst, and David Notkin. Speculative analysis of integrated development environment recommendations. In *OOPSLA 2012, Object-Oriented Programming Systems, Languages, and Applications*, pages 669–682, Tucson, AZ, USA, October 2012.

[82] Sai Zhang, Hao Lü, and Michael D. Ernst. Finding errors in multithreaded GUI applications. In *ISSTA 2012, Proceedings of the 2012 International Symposium on Software Testing and Analysis*, pages 243–253, Minneapolis, MN, USA, July 2012.

[83] Wei Huang, Werner Dietl, Ana Milanova, and Michael D. Ernst. Inference and checking of object ownership. In *ECOOP 2012 — Object-Oriented Programming, 26th European Conference*, pages 181–206, Beijing, China, June 2012.

[84] Werner Dietl, Stephanie Dietzel, Michael D. Ernst, Nathaniel Mote, Brian Walker, Seth Cooper, Timothy Pavlik, and Zoran Popović. Verification games: Making verification fun. In *FTfJP: 14th Workshop on Formal Techniques for Java-like Programs*, pages 42–49, Beijing, China, June 2012.

[85] Eric Spishak, Werner Dietl, and Michael D. Ernst. A type system for regular expressions. In *FTfJP: 14th Workshop on Formal Techniques for Java-like Programs*, pages 20–26, Beijing, China, June 2012.

[86] Jingyue Li and Michael D. Ernst. CBCD: Cloned Buggy Code Detector. In *ICSE 2012, Proceedings of the 34th International Conference on Software Engineering*, pages 310–320, Zürich, Switzerland, June 2012.

[87] Yuriy Brun, Kıvanç Muşlu, Reid Holmes, Michael D. Ernst, and David Notkin. Predicting development trajectories to prevent collaboration conflicts. In *FutureCSD 2012: The Future of Collaborative Software Development*, Bellevue, WA, USA, February 2012.

[88] Colin S. Gordon, Michael D. Ernst, and Dan Grossman. Static lock capabilities for deadlock freedom. In *TLDI 2012: The seventh ACM SIGPLAN Workshop on Types in Language Design and Implementation*, pages 67–78, Philadelphia, PA, USA, January 2012.

[89] Ivan Beschastnikh, Yuriy Brun, Michael D. Ernst, Arvind Krishnamurthy, and Thomas E. Anderson. Bandsaw: Log-powered test scenario generation for distributed systems. In *SOSP WIP: Proceedings of the 23rd ACM Symposium on Operating Systems Principles, Work In Progress Track*, Cascais, Portugal, October 2011.

[90] Ivan Beschastnikh, Yuriy Brun, Michael D. Ernst, Arvind Krishnamurthy, and Thomas E. Anderson. Mining temporal invariants from partially ordered logs. In *SLAML 2011: Workshop on Managing Large-Scale Systems via the Analysis of System Logs and the Application of Machine Learning Techniques (SLAML '11)*, Cascais, Portugal, October 2011. Article No. 3.

[91] Brian Robinson, Michael D. Ernst, Jeff H. Perkins, Vinay Augustine, and Nuo Li. Scaling up automated test generation: Automatically generating maintainable regression unit tests for programs. In *ASE 2011: Proceedings of the 26th Annual International Conference on Automated Software Engineering*, pages 23–32, Lawrence, KS, USA, November 2011.

[92] Sai Zhang, Cheng Zhang, and Michael D. Ernst. Automated documentation inference to explain failed tests. In *ASE 2011: Proceedings of the 26th Annual International Conference on Automated Software Engineering*, pages 63–72, Lawrence, KS, USA, November 2011.

[93] Werner Dietl, Michael D. Ernst, and Peter Müller. Tunable static inference for Generic Universe Types. In *ECOOP 2011 — Object-Oriented Programming, 25th European Conference*, pages 333–357, Lancaster, UK, July 2011.

[94] Sai Zhang, David Saff, Yingyi Bu, and Michael D. Ernst. Combined static and dynamic automated test generation. In *ISSTA 2011, Proceedings of the 2011 International Symposium on Software Testing and Analysis*, pages 353–363, Toronto, Canada, July 2011.

[95] Werner Dietl, Stephanie Dietzel, Michael D. Ernst, Kıvanç Muşlu, and Todd Schiller. Building and using pluggable type-checkers. In *ICSE 2011, Proceedings of the 33rd International Conference on Software Engineering*, pages 681–690, Waikiki, Hawaii, USA, May 2011.

[96] Michael Bayne, Richard Cook, and Michael D. Ernst. Always-available static and dynamic feedback. In *ICSE 2011, Proceedings of the 33rd International Conference on Software Engineering*, pages 521–530, Waikiki, Hawaii, USA, May 2011.

[97] Fausto Spoto and Michael D. Ernst. Inference of field initialization. In *ICSE 2011, Proceedings of the 33rd International Conference on Software Engineering*, pages 231–240, Waikiki, Hawaii, USA, May 2011.

[98] Danny Dig, John Marrero, and Michael D. Ernst. How do programs become more concurrent? a story of program transformations. In *Proceedings of the 4th International Workshop on Multicore Software Engineering*, pages 43–50, Waikiki, Hawaii, USA, May 2011.

[99] Todd W. Schiller and Michael D. Ernst. Rethinking the economics of software engineering. In *FoSER: Workshop on the Future of Software Engineering Research*, pages 325–330, Santa Fe, NM, USA, November 2010.

[100] Yuriy Brun, Reid Holmes, Michael D. Ernst, and David Notkin. Speculative analysis: Exploring future development states of software. In *FoSER: Workshop on the Future of Software Engineering Research*, pages 59–64, Santa Fe, NM, USA, November 2010.

[101] Yoav Zibin, Alex Potanin, Paley Li, Mahmood Ali, and Michael D. Ernst. Ownership and immutability in generic Java. In *OOPSLA 2010, Object-Oriented Programming Systems, Languages, and Applications*, pages 598–617, Revo, NV, USA, October 2010.

[102] Sigurd Schneider, Ivan Beschastnikh, Slava Chernyak, Michael D. Ernst, and Yuriy Brun. Synoptic: Summarizing system logs with refinement. In *SLAML 2010: Workshop on Managing Systems via Log Analysis and Machine Learning Techniques (SLAML '10)*, Vancouver, BC, Canada, October 2010.

[103] Yingyi Bu, Bill Howe, Magdalena Balazinska, and Michael D. Ernst. HaLoop: Efficient iterative data processing on large clusters. In *VLDB 2010: 36th International Conference on Very Large Data Bases*, pages 285–296, Singapore, September 2010.

[104] Jeff H. Perkins, Sunghun Kim, Sam Larsen, Saman Amarasinghe, Jonathan Bachrach, Michael Carbin, Carlos Pacheco, Frank Sherwood, Stelios Sidiroglou, Greg Sullivan, Weng-Fai Wong, Yoav Zibin, Michael D. Ernst, and Martin Rinard. Automatically patching errors in deployed software. In *SOSP 2009, Proceedings of the 22nd ACM Symposium on Operating Systems Principles*, pages 87–102, Big Sky, MT, USA, October 2009.

[105] Adam Kieżun, Philip J. Guo, Karthick Jayaraman, and Michael D. Ernst. Automatic creation of SQL injection and cross-site scripting attacks. In *ICSE 2009, Proceedings of the 31st International Conference on Software Engineering*, pages 199–209, Vancouver, BC, Canada, May 2009.

[106] Danny Dig, John Marrero, and Michael D. Ernst. Refactoring sequential Java code for concurrency via concurrent libraries. In *ICSE 2009, Proceedings of the 31st International Conference on Software Engineering*, pages 397–407, Vancouver, BC, Canada, May 2009.

[107] Matthew M. Papi, Mahmood Ali, Telmo Luis Correa Jr., Jeff H. Perkins, and Michael D. Ernst. Practical pluggable types for Java. In *ISSTA 2008, Proceedings of the 2008 International Symposium on Software Testing and Analysis*, pages 201–212, Seattle, WA, USA, July 2008.

[108] Shay Artzi, Sunghun Kim, and Michael D. Ernst. Recrash: Making software failures reproducible by preserving object states. In *ECOOP 2008 — Object-Oriented Programming, 22nd European Conference*, pages 542–565, Paphos, Cyprus, July 2008.

[109] Jaime Quinonez, Matthew S. Tschantz, and Michael D. Ernst. Inference of reference immutability. In *ECOOP 2008 — Object-Oriented Programming, 22nd European Conference*, pages 616–641, Paphos, Cyprus, July 2008.

[110] Stephen McCamant and Michael D. Ernst. Quantitative information flow as network flow capacity. In *PLDI 2008: Proceedings of the ACM SIGPLAN 2008 Conference on Programming Language Design and Implementation*, pages 193–205, Tucson, AZ, USA, June 2008.

[111] Sunghun Kim and Michael D. Ernst. Which warnings should I fix first? In *ESEC/FSE 2007: Proceedings of the 11th European Software Engineering Conference and the 15th ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 45–54, Dubrovnik, Croatia, September 2007.

[112] Yoav Zibin, Alex Potanin, Mahmood Ali, Shay Artzi, Adam Kieżun, and Michael D. Ernst. Object and reference immutability using Java generics. In *ESEC/FSE 2007: Proceedings of the 11th European Software Engineering Conference and the 15th ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 75–84, Dubrovnik, Croatia, September 2007.

[113] Stephen McCamant and Michael D. Ernst. A simulation-based proof technique for dynamic information flow. In *PLAS 2007: ACM SIGPLAN Workshop on Programming Languages and Analysis for Security*, pages 41–46, San Diego, California, USA, June 2007.

[114] Sunghun Kim and Michael D. Ernst. Prioritizing warnings by analyzing software history. In *MSR 2007: International Workshop on Mining Software Repositories*, pages 27–30, Minneapolis, MN, USA, May 2007.

[115] Carlos Pacheco, Shuvendu K. Lahiri, Michael D. Ernst, and Thomas Ball. Feedback-directed random test generation. In *ICSE 2007, Proceedings of the 29th International Conference on Software Engineering*, pages 75–84, Minneapolis, MN, USA, May 2007.

[116] Adam Kieżun, Michael D. Ernst, Frank Tip, and Robert M. Fuhrer. Refactoring for parameterizing Java classes. In *ICSE 2007, Proceedings of the 29th International Conference on Software Engineering*, pages 437–446, Minneapolis, MN, USA, May 2007.

[117] Michael D. Ernst. The Groupthink specification exercise. In *ICSE Education and Training Track: Software Engineering Education in the Modern Age: Challenges and Possibilities, PostProceedings of ICSE '05 Education and Training Track*, volume 4309 of *Lecture Notes in Computer Science*, pages 89–107. Springer, St. Louis, MO, USA, December 2006.

[118] Shay Artzi, Michael D. Ernst, Adam Kieżun, Carlos Pacheco, and Jeff H. Perkins. Finding the needles in the haystack: Generating legal test inputs for object-oriented programs. In *M-TOOS: 1st Workshop on Model-Based Testing and Object-Oriented Systems*, pages 27–34, Portland, OR, USA, October 2006.

[119] Marcelo d'Amorim, Carlos Pacheco, Darko Marinov, Tao Xie, and Michael D. Ernst. An empirical comparison of automated generation and classification techniques for object-oriented unit testing. In *ASE 2006: Proceedings of the 21st Annual International Conference on Automated Software Engineering*, pages 59–68, Tokyo, Japan, September 2006.

[120] Philip J. Guo, Jeff H. Perkins, Stephen McCamant, and Michael D. Ernst. Dynamic inference of abstract types. In *ISSTA 2006, Proceedings of the 2006 International Symposium on Software Testing and Analysis*, pages 255–265, Portland, ME, USA, July 2006.

[121] Brian Demsky, Michael D. Ernst, Philip J. Guo, Stephen McCamant, Jeff H. Perkins, and Martin Rinard. Inference and enforcement of data structure consistency specifications. In *ISSTA 2006, Proceedings of the 2006 International Symposium on Software Testing and Analysis*, pages 233–243, Portland, ME, USA, July 2006.

[122] Michael D. Ernst, Raimondas Lencevicius, and Jeff H. Perkins. Detection of web service substitutability and composability. In *WS-MaTe: International Workshop on Web Services — Modeling and Testing*, pages 123–135, Palermo, Italy, June 2006.

[123] David Saff, Shay Artzi, Jeff H. Perkins, and Michael D. Ernst. Automatic test factoring for Java. In *ASE 2005: Proceedings of the 20th Annual International Conference on Automated Software Engineering*, pages 114–123, Long Beach, CA, USA, November 2005.

[124] Shay Artzi and Michael D. Ernst. Using predicate fields in a highly flexible industrial control system. In *OOPSLA 2005, Object-Oriented Programming Systems, Languages, and Applications*, pages 319–330, San Diego, CA, USA, October 2005.

[125] Matthew S. Tschantz and Michael D. Ernst. Javari: Adding reference immutability to Java. In *OOPSLA 2005, Object-Oriented Programming Systems, Languages, and Applications*, pages 211–230, San Diego, CA, USA, October 2005.

[126] Michael D. Ernst. Verification for legacy programs. In *VSTTE 2005: Verified Software: Theories, Tools, Experiments*, Zürich, Switzerland, October 2005.

[127] Amy Williams, William Thies, and Michael D. Ernst. Static deadlock detection for Java libraries. In *ECOOP 2005 — Object-Oriented Programming, 19th European Conference*, pages 602–629, Glasgow, Scotland, July 2005.

[128] Carlos Pacheco and Michael D. Ernst. Eclat: Automatic generation and classification of test inputs. In *ECOOP 2005 — Object-Oriented Programming, 19th European Conference*, pages 504–527, Glasgow, Scotland, July 2005.

[129] Jeff H. Perkins and Michael D. Ernst. Efficient incremental algorithms for dynamic detection of likely invariants. In *FSE 2004: Proceedings of the ACM SIGSOFT 12th Symposium on the Foundations of Software Engineering*, pages 23–32, Newport Beach, CA, USA, November 2004.

[130] Stephen McCamant and Michael D. Ernst. Formalizing lightweight verification of software component composition. In *SAVCBS 2004: Specification and Verification of Component-Based Systems*, pages 47–54, Newport Beach, CA, USA, October 2004.

[131] Alan Donovan, Adam Kieżun, Matthew S. Tschantz, and Michael D. Ernst. Converting Java programs to use generic libraries. In *OOPSLA 2004, Object-Oriented Programming Systems, Languages, and Applications*, pages 15–34, Vancouver, BC, Canada, October 2004.

[132] Adrian Birka and Michael D. Ernst. A practical type system and language for reference immutability. In *OOPSLA 2004, Object-Oriented Programming Systems, Languages, and Applications*, pages 35–49, Vancouver, BC, Canada, October 2004.

[133] Lee Lin and Michael D. Ernst. Improving adaptability via program steering. In *ISSTA 2004, Proceedings of the 2004 International Symposium on Software Testing and Analysis*, pages 206–216, Boston, MA, USA, July 2004.

[134] David Saff and Michael D. Ernst. An experimental evaluation of continuous testing during development. In *ISSTA 2004, Proceedings of the 2004 International Symposium on Software Testing and Analysis*, pages 76–85, Boston, MA, USA, July 2004.

[135] Stephen McCamant and Michael D. Ernst. Early identification of incompatibilities in multi-component upgrades. In *ECOOP 2004 — Object-Oriented Programming, 18th European Conference*, pages 440–464, Oslo, Norway, June 2004.

[136] David Saff and Michael D. Ernst. Automatic mock object creation for test factoring. In *PASTE 2004: ACM SIGPLAN/SIGSOFT Workshop on Program Analysis for Software Tools and Engineering (PASTE'04)*, pages 49–51, Washington, DC, USA, June 2004.

[137] Yuriy Brun and Michael D. Ernst. Finding latent code errors via machine learning over program executions. In *ICSE 2004, Proceedings of the 26th International Conference on Software Engineering*, pages 480–490, Edinburgh, Scotland, May 2004.

[138] David Saff and Michael D. Ernst. Continuous testing in Eclipse. In *2nd Eclipse Technology Exchange Workshop (eTX)*, Barcelona, Spain, March 2004.

[139] David Saff and Michael D. Ernst. Reducing wasted development time via continuous testing. In *ISSRE 2003: Fourteenth International Symposium on Software Reliability Engineering*, pages 281–292, Denver, CO, November 2003.

[140] Stephen McCamant and Michael D. Ernst. Predicting problems caused by component upgrades. In *ESEC/FSE 2003: Proceedings of the 9th European Software Engineering Conference and the 11th ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 287–296, Helsinki, Finland, September 2003.

[141] Michael D. Ernst. Static and dynamic analysis: Synergy and duality. In *WODA 2003: Workshop on Dynamic Analysis*, pages 24–27, Portland, OR, USA, May 2003.

[142] Michael Harder, Jeff Mellen, and Michael D. Ernst. Improving test suites via operational abstraction. In *ICSE 2003, Proceedings of the 25th International Conference on Software Engineering*, pages 60–71, Portland, Oregon, May 2003.

[143] Jeremy W. Nimmer and Michael D. Ernst. Invariant inference for static checking: An empirical evaluation. In *FSE 2002, Proceedings of the ACM SIGSOFT 10th International Symposium on the Foundations of Software Engineering*, pages 11–20, Charleston, SC, November 2002.

[144] Jeremy W. Nimmer and Michael D. Ernst. Automatic generation of program specifications. In *ISSTA 2002, Proceedings of the 2002 International Symposium on Software Testing and Analysis*, pages 232–242, Rome, Italy, July 2002.

[145] Yoshio Kataoka, Michael D. Ernst, William G. Griswold, and David Notkin. Automated support for program refactoring using invariants. In *ICSM 2001: Proceedings of the International Conference on Software Maintenance*, pages 736–743, Florence, Italy, November 2001.

[146] Jeremy W. Nimmer and Michael D. Ernst. Static verification of dynamically detected program invariants: Integrating Daikon and ESC/Java. In *RV 2001: Proceedings of the First Workshop on Runtime Verification*, Paris, France, July 2001.

[147] Michael D. Ernst, Adam Czeisler, William G. Griswold, and David Notkin. Quickly detecting relevant program invariants. In *ICSE 2000, Proceedings of the 22nd International Conference on Software Engineering*, pages 449–458, Limerick, Ireland, June 2000.

[148] Michael D. Ernst, Craig S. Kaplan, and Craig Chambers. Predicate dispatching: A unified theory of dispatch. In *ECOOP '98: the 12th European Conference on Object-Oriented Programming*, pages 186–211, Brussels, Belgium, July 1998.

[149] Michael D. Ernst, Todd D. Millstein, and Daniel S. Weld. Automatic SAT-compilation of planning problems. In *IJCAI '97: IJCAI-97, Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 1169–1176, Nagoya, Aichi, Japan, August 1997.

[150] Daniel Weise, Roger F. Crew, Michael D. Ernst, and Bjarne Steensgaard. Value dependence graphs: Representation without taxation. In *POPL '94: Proceedings of the 21st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 297–310, Portland, OR, January 1994.

[151] Michael D. Ernst and Bruce E. Flinchbaugh. Image/map correspondence using curve matching. In *AAAI Spring Symposium on Robot Navigation*, Stanford, CA, March 28–30, 1989. Also published as Texas Instruments Technical Report CSC-SIUL-89-12.

[152] Michael D. Ernst. Ml typechecking is not efficient. In *Papers of the MIT ACM Undergraduate Conference*, April 1989.

Conference proceedings and other non-journal articles – Refereed by abstract only Not applicable.

Complete books written Not applicable.

Parts of books (chapters in edited books) Not applicable.

Books edited Not applicable.

Proceedings and journal issues edited

[153] Michael D. Ernst and Thomas Jensen, editors. *PASTE: ACM SIGPLAN/SIGSOFT Workshop on Program Analysis for Software Tools and Engineering*, Lisbon, Portugal, September 2005.

[154] Jonathan E. Cook and Michael D. Ernst, editors. *WODA 2003: ICSE Workshop on Dynamic Analysis*, Portland, Oregon, May 2003.

[155] Michael D. Ernst, editor. *IR '95: Intermediate Representations Workshop Proceedings*, San Francisco, CA, January 22, 1995. *ACM SIGPLAN Notices* 30(3), March 1995.

[156] Michael D. Ernst. Intellectual property in computing: (How) should software be protected? An industry perspective. Memo AIM-1369, MIT Artificial Intelligence Laboratory, Cambridge, Massachusetts, May 1992.

Patents submitted and/or awarded

[157] Gideon A. Yuval and Michael D. Ernst. Method and system for controlling unauthorized access to information distributed to users. U.S. Patent 5,586,186, December 17, 1996. Assigned to Microsoft Corporation.

Abstracts, letters, non-refereed papers, technical reports

[158] Michael D. Ernst. Type Annotations specification (JSR 308). `https://checkerframework.org/jsr308/`, October 2011.

[159] David Saff, Marat Boshernitsan, and Michael D. Ernst. Theories in practice: Easy-to-write specifications that catch bugs. Technical Report MIT-CSAIL-TR-2008-002, MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, January 14, 2008.

[160] Michael D. Ernst and Jeff H. Perkins. Learning from executions: Dynamic analysis for software engineering and program understanding, November 2005. Tutorial at ASE 2005.

[161] Nii Dodoo, Lee Lin, and Michael D. Ernst. Selecting, refining, and evaluating predicates for program analysis. Technical Report MIT-LCS-TR-914, MIT Laboratory for Computer Science, Cambridge, MA, July 21, 2003.

[162] Samir V. Meghani and Michael D. Ernst. Determining legal method call sequences in object interfaces. `https://homes.cs.washington.edu/~mernst/pubs/call-sequences.pdf`, May 2003.

[163] David Notkin, Marc Donner, Michael D. Ernst, Michael Gorlick, and E. James Whitehead, Jr. Panel: Perspectives on software engineering. In *ICSE 2001, Proceedings of the 23rd International Conference on Software Engineering*, pages 699–702, Montreal, Canada, May 2001.

[164] Michael D. Ernst. *Dynamically Discovering Likely Program Invariants*. PhD thesis, University of Washington Department of Computer Science and Engineering, Seattle, Washington, August 2000.

[165] Michael D. Ernst, William G. Griswold, Yoshio Kataoka, and David Notkin. Dynamically discovering pointer-based program invariants. Technical Report UW-CSE-99-11-02, University of Washington Department of Computer Science and Engineering, Seattle, WA, November 16, 1999. Revised March 17, 2000.

[166] Michael D. Ernst. Slicing pointers and procedures (abstract). Technical Report MSR-TR-95-23, Microsoft Research, Redmond, WA, January 13, 1995.

[167] Michael D. Ernst. Serializing parallel programs by removing redundant computation. Technical Report MIT/LCS/TR-638, MIT Laboratory for Computer Science, Cambridge, MA, August 21, 1994.

[168] Michael D. Ernst. Practical fine-grained static slicing of optimized code. Technical Report MSR-TR-94-14, Microsoft Research, Redmond, WA, July 26, 1994.

[169] Michael D. Ernst and Gideon Yuval. Heraclitean encryption. Technical Report MSR-TR-94-13, Microsoft Research, Redmond, WA, March 3, 1994.

[170] Michael D. Ernst. Adequate models for recursive program schemes. Bachelors thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1989.

Other significant research dissemination (web sites, software, Wikis, etc.)

In order to permit others to reproduce my results and build upon my research, I make all my research artifacts publicly available (most linked from `https://homes.cs.washington.edu/~mernst/software/`, along with more information about each; and others by request). Some of the more significant research systems that are used by others include the following. Most of these were created in collaboration with my students and other colleagues.

1. Daikon dynamically detects likely program invariants in C, C++, Java, Perl, and other languages. Almost 100 papers (that I know of; see `http://plse.cs.washington.edu/daikon/pubs/#daikon-methodology`) use Daikon as an integral part of their research methodology, and additional papers use Daikon as a test subject Daikon is being used internally by a number of companies, and is being commercialized by Agitar, Determina, and Scrutiny (that I know of). Regarding Agitar's outgrowth from Daikon, see their ISSTA 2006 paper "From Daikon to Agitator: Lessons and Challenges in Building a Commercial Tool for Developer Testing". Agitar's awards include Java One Duke's Choice, Jolt award, the Wall Street Journal Software Technology Innovation Award, and InfoWorld Technology of the Year.
2. The generics refactoring in Eclipse — the most widely-used Java integrated development environment (IDE) — is an adaptation of our sound, precise, and scalable systems that introduce parametric polymorphism (generic types) in Java programs.

3. Continuous testing augments an IDE to rapidly indicate semantic errors, just as they already do for syntactic errors. The key idea is to run tests in the background and indicate test failures. Careful user interface design results in a system that helps — not annoys — programmers, as indicated by case studies and controlled experiments. Continuous testing is a popular plug-in for Eclipse.
4. Version 4 of the popular and widely-used JUnit regression testing framework was built in part in my group, by David Saff (in conjunction with Kent Beck and Erich Gamma). The JUnit plug-in distributed with Eclipse (version 3.2 and later) was built in my group, by David Saff and others. It replaced a previous version built by the Eclipse team that had lesser functionality.
5. Test factoring is a capture–replay technique for converting a long-running system test into a collection of fast unit tests that exercise software components in the same way the system test did. Our implementation scales to programs of hundreds of thousands of lines, and others have since produced their own implementations.
6. The Eclat, Joe, Randoop, and Palulu systems automatically generate test inputs, using novel techniques to create relevant and effective tests. Collectively, these four systems have found hundreds of errors in the Sun and IBM Java Development Kits (JDKs), and in the Microsoft equivalent (the Common Language Runtime), which are heavily-tested, widely-deployed commercial infrastructure.
7. The Checker Framework allows easy creation of pluggable type-checkers. A pluggable type-checker is a compiler plug-in that extends and strengthens the type system of a programming language. As a result, the compiler can detect and prevent errors that would otherwise have caused incorrect behavior at run time. We built over a dozen type-checkers using this framework, and research groups around the world have adopted the Checker Framework to build many more. This work was inspired by the difficulty we experienced when creating specialized type-checkers, such as that we built for Javari.
8. Synoptic mines system logs and produces a model of observed system behavior, to help programmers to understand their systems and fix bugs, improve performance, or perform maintenance.
9. Eclipse's Quick Fix facility suggests ways to fix a compilation errors. Kıvanç Muşlu's Quick Fix Scout plug-in augments the Quick Fix menu by indicating the effect of each suggestion (how many compilation errors it fixes), adding better suggestions, and sorting all the suggestions. It does this by actually trying each suggestion in the background.
10. The Crystal tool proactively warns when two developers make conflicting changes to a codebase, before the developers have shared their changes with one another. By utilizing local commits and speculative analysis, it suffers no false positives, unlike other tools. Microsoft has re-implemented this tool as Beacon, and is deploying Beacon internally.
11. A static ownership inference system (for Generic Universe Types) enables tuning by users to find the best among many possible legal ownership structures, using a Max-SAT solver.
12. The Ductile system combines the benefits of static and dynamic typing, allowing a programmer to utilize sound static typing or fully dynamic execution, using the same codebase and switching between the two on demand with no program edits required.
13. HaLoop makes the popular Hadoop Map–Reduce implementation more efficient, by caching and re-using results in iterative computations.
14. The HAMPI solver for string constraints is used in a variety of security and verification contexts.
15. The ClearView system automatically detects security attacks and bugs, proposes and evaluates fixes that inoculate the system from the attacks/bugs, and deploys the best fix — all without human intervention and in seconds.
16. We have extended Java's annotation system, and Oracle plans to include our extension (in Java parlance, "JSR 308") in the Java 8 language and use our implementation in the Java Development Kit (JDK).

17. The Fjalar toolkit enables instrumentation of compiled executables (currently for Linux/x86). It combines the benefits of binary and source rewriting through a "mixed-level" approach. It is the basis for a value profiling tool (Kvasir) and an abstract type inference (DynComp).
18. DynComp infers abstract types for Java, C, and C++ programs, retrieving structure from source code in which a single type is used to represent multiple concepts.
19. The Groupthink Specification Exercise is a fun group activity that teaches students the value of specifications, the difficulty of creating them, and various approaches for doing so. It has been used worldwide. I distribute, handouts, lecture slides, instructors' notes, and optional software that automates running the activity.
20. Stephen McCamant's PittSFIeld tool performs efficient sandboxing for the x86 architecture, which presents challenges (such as variable-length instructions) over RISC systems for which sandboxing had been previously applied.
21. Medic was the first tool for compiling a problem into SAT, or logical satisfiability. (The idea was due to Kautz and Selman, but this is the first implementation.) Such a transformation is valuable because of engineering and research that has made SAT-solving very fast. Today, solving problems by reducing them to SAT is a very common technique used in all fields of computer science.
22. The Gamesman's Toolkit facilitates combinatorial game-theoretic analysis of complete-information games. I extended it with support for analyzing the ancient Hawaiian game of Konane.
23. Subject programs from my experiments are often requested by other researchers who do not wish to go to the trouble of collecting and organizing realistic programs that can be used to validate research in testing and other areas of software engineering.

Other software systems stemming from my research that are used by others include cppp (a partial evaluator for cpp, the C preprocessor), Gud (a prototype implementation of predicate dispatching), and contributions to the IOA toolkit distributed by Nancy Lynch's Theory of Distributed Systems group. Educational contributions include infrastructure for supporting programming classes. Widely used software not stemming from research includes EDB (a database system), contributions to Emacs and other free software, bibtex2web (software for creating web pages from BibTeX bibliography files), and others. I also maintain a popular collection of advice for students and researchers, at `http://homes.cs.washington.edu/~mernst/advice/`.

OTHER SCHOLARLY ACTIVITY

Invited lectures and seminars

Only talks since 2000 are listed. Talks at closed events (e.g., PI meetings, UW events) are not listed. Talks given in connection with a refereed publication are not listed. Talks given by co-authors are not listed.

1. SCAM, "Custom type-checking for programming, teaching, and research", September 28, 2020
2. Amazon Web Services, "Implement your own type system in 1 hour", March 20, 2019
3. ISSTA, "Pluggable type systems reconsidered", Impact Paper Award talk, July 2018
4. Code One "Implement Your Own Type System in 45 Minutes", October 2018
5. Code One "Preventing Errors Before They Happen: The Checker Framework", October 2018
6. Code One "Using Type Annotations to Improve Your Code", October 2018

7. ISSTA, “Comparing developer-provided to user-provided tests for fault localization and automated program repair”, July 2018
8. Oracle, “Create your own type system in 1 hour”, May 16, 2018
9. Seattle Java User’s Group, “Create your own type system in 1 hour”, February 20, 2018
10. JavaOne, “Preventing Errors Before They Happen”, Oct 2, 2017
11. JavaOne, “Using Type Annotations to Improve Your Code”, Oct 2, 2017
12. Open Source Bridge, “Create your own type system in 45 minutes”, June 22, 2017
13. ETAPS keynote, “Natural language is a programming language”, April 27, 2017
14. TU Delft, “Preventing null pointer exceptions at compile time”, April 25, 2017
15. TU Delft, “Natural language is a programming language”, April 25, 2017
16. CWI Amsterdam (Centrum Wiskunde & Informatica) “Preventing errors before they happen”, April 24, 2017
17. Devoxx, “Preventing Null pointer exceptions at compile time”, March 22, 2017
18. Devoxx, “The Checker Framework in action: Preventing errors kefore they happen”, March 21, 2017
19. Georgia Institute of Technology, “Natural language is a programming language”, January 24, 2017
20. Triangle Computer Science Distinguished Lecturer Series, “Natural language is a programming language”, January 23, 2017
21. Microsoft Research, “Analyzing the entire program: applying natural language processing to software engineering”, January 11, 2017
22. University of Southern California, “Analyzing the entire program: applying natural language processing to software engineering”, December 1, 2016
23. JavaOne, “Preventing Errors Before They Happen”, September 19, 2016
24. JavaOne, “Disciplined Locking: No More Concurrency Errors”, September 19, 2016
25. JavaOne, “Using Type Annotations to Improve Your Code”, September 19, 2016
26. Pacific Northwest Programming Languages and Software Engineering Meeting, “Natural language processing meets software testing”, March 15, 2016
27. University of Washington at Bothell, “Preventing errors before they happen with pluggable type-checking”, November 16, 2015
28. JavaOne, “Preventing Errors Before They Happen”, October 26, 2015
29. JavaOne, “Collaborative Verification of the Information Flow for a High-Assurance App Store”, October 26, 2015
30. JavaOne, “Using Type Annotations to Improve Your Code”, October 26, 2015
31. University of Buenos Aires, “Preventing errors before they happen: Lightweight verification via pluggable type-checking”, June 5, 2015
32. University of Buenos Aires, “Collaborative verification of information flow for a high-assurance app store”, June 1, 2015
33. CSI-SE, (keynote), May 19, 2015
34. CIbSE, “Lightweight software verification with pluggable type-checking”, April 23, 2015
35. University of Buenos Aires, “Verification games: Making software verification fun”, April 13, 2015
36. IMDEA Software Institute, “Collaborative verification of information flow for a high-assurance app store”, November 18, 2014
37. Jazoon, “GUI threading explained and verified”, October 21, 2014
38. Coverity, “The Checker Framework: Pluggable static analysis for Java”, August 27, 2014
39. SIGCSE, “Introductory programming meets the real world: Using real problems and data in CS1”, March 7, 2014

40. Georgia Institute of Technology, "Verification games: Making software verification fun", March 6, 2014
41. University of California at San Diego, "Collaborative verification of information flow for a high-assurance app store", January 28, 2014
42. University of Texas, "Verification games: Making software verification fun", December 2, 2013
43. University of Texas, "Collaborative verification of information flow for a high-assurance app store", December 2, 2013
44. University of Wisconsin Distinguished Lecture, "Verification games: Making software verification fun", November 13, 2013
45. University of British Columbia ECE Distinguished Lecture, "Verification games: Making software verification fun", September 9, 2013
46. JavaOne, "Enhanced Metadata in Java SE 8", September 23, 2012
47. FuSE, "All work and no play makes software engineering dull" (keynote), July 17, 2013
48. Notkinfest, "Software evolution then and now: The research impact of David Notkin", February 1, 2013
49. JavaOne, "Build Your Own Type System for Fun and Profit", October 2, 2012
50. OSCON, "Preventing Runtime Errors at Compile Time", July 17, 2012
51. ICST, "Reproducible Tests? Non-duplicable Results in Testing and Verification" (keynote), April 20, 2012
52. Oracle, "Detecting and preventing bugs with pluggable type-checking", April 2012
53. Oracle, "JSR 308: Type Annotations: Making Java annotations more general and more useful", January 10, 2012
54. University of Maryland, "Always-available static and dynamic feedback: Unifying static and dynamic typing", November 8, 2011
55. University of Maryland, "Verification games: Crowd-sourced program verification", November 8, 2011
56. ICSE, "Always-available static and dynamic feedback", May 27, 2011
57. Oracle, "Detecting and preventing bugs with pluggable type-checking", March 2011
58. Usable Verification workshop, "User-specified type systems for domain-specific verification", Nov 16, 2010
59. Microsoft, RiSE all-hands meeting, August 5, 2010
60. TAP keynote, "How tests and proofs impede one another: The need for always-on static and dynamic feedback", July 1, 2010
61. Devoxx, "Detecting and preventing bugs with pluggable type-checking", November 2009
62. OOPSLA tutorial, "Preventing bugs with pluggable type checking", October 2009
63. University of Washington, "Preventing bugs with pluggable type checking", October 2009
64. OOPSLA tutorial, "Preventing bugs with pluggable type checking", October 26, 2009
65. SCAM keynote, "How analysis can hinder source code manipulation – and what to do about it", Sep 20, 2009
66. PPPJ tutorial, "Preventing bugs with pluggable type checking", Aug 27, 2009
67. IBM, "Self-defending software: Automatically patching security vulnerabilities", Aug 25, 2009
68. Microsoft, RiSE all-hands meeting, July 1, 2009
69. Microsoft, "Towards better software: Finding, fixing, and preventing errors", June 29, 2009
70. JavaOne, "Preventing bugs with pluggable type checking", June 2009
71. Seattle Java User's Group, "Preventing bugs with pluggable type checking", April 2009
72. Microsoft Research, RiSE Group, "Making it easier (and more fun!) to create software", April 21,

2009

73. Javoxx, "Preventing bugs with pluggable type-checking for Java", December 2008
74. Google, "Preventing bugs with pluggable type checking", December 2008
75. MPI-SWS Distinguished Lecture, "Self-defending software: Automatically patching security vulnerabilities", Nov 25, 2008
76. University of Kaiserslautern, "Self-defending software: Automatically patching security vulnerabilities", Nov 24, 2008
77. University of Karlsruhe, "Self-defending software: Automatically patching security vulnerabilities", Nov 17, 2008
78. King's College London, CREST center, "Self-defending software: Automatically patching security vulnerabilities", Nov 12, 2008
79. OOPSLA, "Enforcing reference and object immutability in Java", October 2008
80. OOPSLA, "Compile-time type-checking for custom type qualifiers in Java", October 2008
81. ISSTA, "Practical pluggable types for Java", July 23, 2008
82. Max Planck Institute for Software Systems, "Practical pluggable types for Java", July 17, 2008
83. ECOOP, "ReCrash: Making software failures reproducible by preserving object states", July 11, 2008
84. ECOOP, "Inference of reference immutability", July 11, 2008
85. ECOOP, "Building and using pluggable type systems with the Checker Framework", July 10, 2008
86. University of Saarland, "ReCrash: Making software failures reproducible by preserving object states", July 8, 2008
87. University of Saarland, "Inference of reference immutability", July 4, 2008
88. JavaOne, "Upcoming Java programming-language changes", May 6, 2008 (with Alex Buckley)
89. Dagstuhl workshop Scalable Program Analysis, "Scalable pluggable types", April 14–18, 2008
90. J-Spring, "Preventing bugs with pluggable type-checking for Java", April 16, 2008
91. University of Washington, "Self-defending software: Collaborative learning for security", April 1, 2008
92. QCon, "User-defined type systems for error detection and prevention", November 9, 2007
93. OOPSLA, "Compile-time typechecking for custom Java type qualifiers", October 2007
94. Harvard University, "Refactoring for parameterizing Java classes", March 7, 2007
95. Stanford University, "Refactoring for parameterizing Java classes", January 8, 2007
96. Microsoft Research, "Dynamic inference of abstract types", December 12, 2006
97. University of California at Berkeley, "Combined static and dynamic mutability analysis", December 11, 2006
98. IBM T.J. Watson Research Center, "Combined static and dynamic mutability analysis", November 28, 2006
99. Microsoft Research India, "Evaluating systematic and random testing", October 10, 2006
100. Tata Research Development & Design Centre, "Combined static and dynamic mutability analysis", September 29, 2006
101. Indian Institute of Technology, Delhi, "Evaluating systematic and random testing", September 28, 2006
102. IBM India Research Lab, "Choosing the right tests: test selection via operational abstraction", September 28, 2006
103. Harvard University, "Javari: Adding reference immutability to Java", February 22, 2006
104. University of California at San Diego, "Javari: Adding reference immutability to Java", November 8, 2005
105. Tutorial at Conference on Automated Software Engineering 2005, "Learning from executions: Dynamic

analysis for program understanding and software engineering", November 7, 2005

106. University of Arizona, "Eclat: Automatic generation and classification of test inputs", October 27, 2005
107. Conference on Verified Software: Theories, Tools, Experiments, "Verification for legacy programs", October 10, 2005
108. DARPA Self-Regenerative Systems meeting, "Learning and repair techniques for self-healing systems", July 12, 2005
109. ABB, "Making the most of impoverished test suites: Automatic classification of test inputs and test factoring", November 19, 2004
110. OOPSLA Workshop on the Java Platform: Tiger and Beyond, "Integrating reference immutability into the Java mainstream", October 28, 2004
111. Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA), "A practical type system and language for reference immutability", October 26, 2004
112. Philips Medical, "Making the most of impoverished test suites: Automatic classification of test inputs and test factoring", October 4, 2004
113. UW/MSR Summer Institute on Trends in Testing: Theory, Techniques and Tools (keynote), "Making the most of impoverished test suites: Automatic classification of test inputs and test factoring", August 24, 2004
114. Program Analysis and Software Tools for Engineering (PASTE) workshop (keynote), "Static and dynamic analysis: Synergy and duality", June 7, 2004
115. Carnegie Mellon University, "Improving adaptability via program steering", April 27, 2004
116. University of Limerick, "Using dynamic analysis to assist theorem proving", December 12, 2003
117. Cambridge University and Microsoft Research, "Predicting problems caused by component upgrades", December 10, 2003
118. Dagstuhl workshop Understanding Program Dynamics, "Comparing dynamic program behaviors", December 2, 2003
119. Dagstuhl workshop Understanding Program Dynamics (keynote), "Static and dynamic analysis: Synergy and duality", December 1, 2003
120. IBM T.J. Watson Research Center (Distinguished Lecture), "Improving test suites via operational abstraction", June 16, 2003
121. ICSE Workshop on Dynamic Analysis (WODA), Portland, Oregon, "Static and dynamic analysis: Synergy and duality", May 9, 2003
122. International Conference on Software Engineering (ICSE), Portland, Oregon, "Improving test suites via operational abstraction", May 6, 2003
123. Microsoft Research, Portland, Oregon, "Improving test suites via operational abstraction", May 2, 2003
124. University of Pittsburgh, "Predicting problems caused by component upgrades", February 4, 2003
125. Carnegie Mellon University, "Improving test suites via operational abstraction", February 3, 2003
126. Cambridge University, England, "Using dynamic analysis to assist program verification", July 25, 2002
127. Java Verification Workshop (JVW'01), Portland, Oregon, "Static verification of dynamically detected program invariants", January 13, 2002
128. McMaster University, "Refactoring and static verification: Two applications of dynamic invariant detection", January 25, 2002
129. Rice University, "Refactoring and static verification: Two applications of dynamic invariant detection", January 15, 2002
130. University of Washington, "Refactoring and static verification: Two applications of dynamic invariant

detection", January 10, 2002

131. Williams College, "Refactoring and static verification: Two applications of dynamic invariant detection", November 30, 2001
132. International Conference on Software Maintenance (ICSM), Florence, Italy, "Automated Support for Program Refactoring using Invariants", November 9, 2001
133. International Conference on Software Maintenance (ICSM), Florence, Italy, "Dynamically Detecting Likely Program Invariants", November 8, 2001
134. IBM T.J. Watson Research Center, "Refactoring and static verification: Two applications of dynamic invariant detection", August 8, 2001
135. Brunel University, "Overview of dynamic invariant detection", July 24, 2001
136. Workshop on Runtime Verification (RV'01), Paris, France, "Static verification of dynamically detected program invariants", July 23, 2001
137. International Conference on Software Engineering (ICSE), Montreal, Canada, "The future of software engineering" (panel), May 18, 2001
138. University of Wisconsin at Madison, "Refactoring and static verification: Two applications of dynamic invariant detection", May 14, 2001
139. University of California at Irvine, "Dynamically Detecting Likely Program Invariants", February 23, 2001
140. Compaq Systems Research Center, "Dynamically Detecting Likely Program Invariants", February 22, 2001
141. University of Southern California, "Dynamically Detecting Likely Program Invariants", February 21, 2001
142. MIT, "Playing Konane Mathematically with Combinatorial Game Theory", January 17, 2001
143. Brown University, "Dynamically Detecting Likely Program Invariants", February 1, 2001
144. Tufts University, "Dynamically Detecting Likely Program Invariants", November 20, 2000
145. University of Virginia, "Top Gun" Distinguished Lecture Series, "Dynamically Detecting Likely Program Invariants", November 13, 2000
146. Boston University, "Dynamically Detecting Likely Program Invariants", September 25, 2000
147. Massachusetts Institute of Technology, "Dynamically Detecting Likely Program Invariants", April 10, 2000
148. University of Maryland, "Dynamically Detecting Likely Program Invariants", April 5, 2000
149. Carnegie Mellon University, "Dynamically Detecting Likely Program Invariants", April 3, 2000
150. University of Massachusetts at Amherst, "Dynamically Detecting Likely Program Invariants", March 29, 2000
151. Georgia Institute of Technology, "Dynamically Detecting Likely Program Invariants", March 23, 2000
152. Stanford University, "Dynamically Detecting Likely Program Invariants", March 6, 2000
153. University of California at Berkeley, "Dynamically Detecting Likely Program Invariants", March 2, 2000
154. University of California at San Diego, "Dynamically Detecting Likely Program Invariants", February 29, 2000
155. University of Colorado at Boulder, "Dynamically Detecting Likely Program Invariants", February 24, 2000
156. Cornell University, "Dynamically Detecting Likely Program Invariants", February 15, 2000

Presentations given at conferences

All of the conference proceedings articles listed earlier (page 5), plus others that were superseded by a later publication and so do not appear on this cv, were presented at a conference. Most were given by my coauthors, such as students, and I do not list those here.

Professional society memberships

Association for Computing Machinery (ACM) since 1996
Eta Kappa Nu (electrical engineering honor society) since 1987
Institute for Electical and Electronics Engineering (IEEE) since 2013
Sigma Xi (scientific research honor society) since 1987
Tau Beta Pi (engineering honor society) since 1987

Program committees and journal editorships

1. BenchWork 2019.
2. SecDev 2017. Conference on Secure Development.
3. FSE 2016 SRC. Student Research Competition Committee, ACM SIGSOFT International Symposium on the Foundations of Software Engineering.
4. SecDev 2016. Conference on Secure Development.
5. PLDI 2016 EPC. PLDI 2016 DPC. ACM Conference on Programming Language Design and Implementation, External Program Committee and Distinguished Paper Committee.
6. OOPSLA 2016 ERC. Conference on Object-Oriented Programming Systems, Languages and Applications, External Review Committee.
7. ICSE 2016. 38th International Conference on Software Engineering.
8. ESEC/FSE 2015 DS. Doctoral Symposium of the 11th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering.
9. GAS 2015. 4th International Workshop on Games and Software Engineering.
10. PLDI 2014 ERC. ACM Conference on Programming Language Design and Implementation, External Review Committee.
11. ACM SIGSOFT Dissertation Award 2014.
12. ESEC/FSE 2013. 9th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering.
13. Co-chair, FuSE 2013. Future of Software Engineering symposium.
14. Co-chair, ICSE 2013 tutorials. 35th International Conference on Software Engineering.
15. ICSE 2013 Mentoring Committee. 35th International Conference on Software Engineering.
16. WODA 2013. 11th International Workshop on Dynamic Analysis.
17. Chair, SPLASH 2012 Panels. Systems, Programming, Languages and Applications: Software for Humanity.
18. SPLASH 2012 Demos. Systems, Programming, Languages and Applications: Software for Humanity.
19. SPLASH 2012 Doctoral symposium. Systems, Programming, Languages and Applications: Software for Humanity.

20. OOPSLA 2011. Conference on Object-Oriented Programming Systems, Languages and Applications
21. ICSE 2011. 33rd International Conference on Software Engineering
22. TAIC PART 2010. Testing: Academic and Industrial Conference - Practice and Research Techniques
23. WODA 2009. Workshop on Dynamic Analysis,
24. ICSE 2009 Mentor Program. 31st International Conference on Software Engineering
25. WODA 2009. Workshop on Dynamic Analysis
26. ICSE 2009 Mentor Program. 31st International Conference on Software Engineering
27. HotSWUp 2008. ACM Workshop on Hot Topics in Software Upgrades
28. ISSTA 2008. International Symposium on Software Testing and Analysis
29. ESEC-FSE 2007. European Software Engineering Conference and ACM International Symposium on the Foundations of Software Engineering
30. TAP 2007. Tests and Proofs
31. ICSE 2007 Education Track. 29th International Conference on Software Engineering
32. eTX 2006. Eclipse Technology Exchange Workshop
33. ECOOP 2006. European Conference on Object-Oriented Programming
34. ISSTA 2006. International Symposium on Software Testing and Analysis
35. PLDI 2006. ACM Conference on Programming Language Design and Implementation
36. WODA 2006. Workshop on Dynamic Analysis
37. CC 2006. International Conference on Compiler Construction
38. VSTTE 2005. Verified Software: Theories, Tools, Experiments
39. Co-chair, PASTE 2005. ACM Workshop on Program Analysis for Software Tools and Engineering
40. ESDDT 2005 ("Bugs 2005"). Workshop on the Evaluation of Software Defect Detection Tools
41. POPL 2005. ACM Symposium on Principles of Programming Languages
42. FTfJP 2004. Formal Techniques for Java-like Programs Workshop
43. PASTE 2004. ACM Workshop on Program Analysis for Software Tools and Engineering
44. WODA 2004. Workshop on Dynamic Analysis
45. ICSE 2004. 26th International Conference on Software Engineering
46. PL Day 2004. IBM Programming Language Day Workshop
47. eTX 2004. Eclipse Technology Exchange Workshop
48. Co-chair, WODA 2003. Workshop on Dynamic Analysis
49. PASTE 2002. ACM Workshop on Program Analysis for Software Tools and Engineering
50. FSE 2002. ACM Tenth International Symposium on the Foundations of Software Engineering
51. RV 2002. Second Workshop on Runtime Verification
52. NEPLS 6. Sixth New England Programming Languages and Systems Symposium, 2002
53. Chair, NEPLS 5. Fifth New England Programming Languages and Systems Symposium, 2002
54. SCAM 2001. IEEE International Workshop on Source Code Analysis and Manipulation
55. NEPLS 4. Fourth New England Programming Languages and Systems Symposium, 2001
56. RV 2001. Workshop on Runtime Verification
57. Chair, IR 1995. ACM SIGPLAN Intermediate Representations Workshop

In addition, I have been an external reviewer for too many additional conferences to count, and likewise have done many journal reviews. Here is a rather incomplete list, as of 2006: Computer-aided Verification (CAV), Empirical Software Engineering Journal, Foundations of Software Engineering (FSE), International Conference on Software Engineering (ICSE) education track, International Joint Conference on Artificial Intelligence (IJCAI), International Journal of Software and Informatics (IJSI), International Symposium on Software Testing and Analysis (ISSTA), ACM SIGPLAN International Conference on Object-Oriented

Programming, Systems, Languages, and Applications (OOPSLA), ACM Conference on Programming Language Design and Implementation (PLDI), ACM Symposium on Principles of Programming Languages (POPL), ACM European SigOps Workshop, ACM Symposium on Operating System Principles (SOSP), International Workshop on Test and Analysis of Component Based Systems (TACoS), ACM Transactions on Programming Languages and Systems (TOPLAS), ACM Transactions on Software Engineering and Methodology (TOSEM), IEEE Transactions on Dependable and Secure Computing (TDSC), IEEE Transactions on Software Engineering (TSE).

GRADUATE STUDENTS

Chaired Doctoral Degrees

1. Jordan Samhi, *Analyzing the Unanalyzable: an Application to Android Apps*, Université du Luxembourg, January 2023 (co-advised with Jacques Klein and Tegawende Bissyande).
2. Martin Kellogg, *Lightweight Verification via Specialized Typecheckers*, June 2022. Current employer: New Jersey Institute of Technology.
3. Doug Woos, *A step-through debugger for distributed systems*, 2019 (co-advised with Tom Anderson and Zach Tatlock). Current employer: Brown University.
4. Pavel Panchekha, *Automated Reasoning for Web Page Layout*, 2019 (co-advised with Zach Tatlock). Current employer: University of Utah.
5. Calvin Loncaric, *Data Structure Synthesis*, 2018. Current employer: Oracle.
6. Stuart Pernsteiner, *Practical Verification of Safety-Critical Systems*, 2018 (co-advised with Zach Tatlock). Current employer: Galois.
7. Alberto Goffi, *Automating Test Oracles Generation* 2018 (co-advised with Mauro Pezzè).
8. Konstantin Weitz, *Formal semantics and scalable verification for the Border Gateway Protocol using proof assistants and SMT solvers*, March 2017, co-advised with Zach Tatlock. Current employer: Google.
9. Brian Burg, *Understanding dynamic behavior with tools for retroactive investigation*, June 2015, co-advised with Andrew Ko. Current employer: Apple.
10. Kıvanç Muşlu, *Enhancing software development techniques via copy codebases*, June 2015, co-advised with Yuriy Brun. Current employer: Microsoft.
11. Sai Zhang, *Effective program analyses for automated software testing and error diagnosis*, August 2014.
12. Colin Gordon, *Verifying concurrent programs by controlling alias interference*, July 2014, co-advised with Daniel Grossman. Current employer: Drexel University.
13. Todd Schiller, *Reducing the usability barrier to specification and verification*, June 2014. Current employer: Bridgewater Associates.
14. Ivan Beschastnikh, *Modeling Systems from Logs of their Behavior*, June 2013, co-advised with Thomas Anderson and Arvind Krishnamurthi. Current employer: University of British Columbia.
15. Adam Kieżun, *Effective software testing with a string-constraint solver*, May 2009. Current employer: Harvard University.
16. Shay Artzi, *Dynamically fighting bugs: Prevention, detection, and elimination*, June 2009. Current employer: IBM.

17. Stephen McCamant, *Quantitative information-flow tracking for real systems*, May 2008. Current employer: University of Minnesota.

Current Doctoral Students

Student Name, indicate Chair or Co-chair (include name of co-chair, if applicable), Status (e.g. achieved candidacy, year; or passed qualifying exam, year)

1. Calvin Loncaric
2. Pavel Panchekha
3. Stuart Pernsteiner
4. Doug Woos

Chaired Masters Degrees

M. S. Theses:

S. M. Theses:

1. Carlos Pacheco, S.M. Thesis: *Eclat: Automatic generation and classification of test inputs*, June 2005.
2. Amy Williams, S.M. Thesis *Static detection of deadlock for Java libraries*, May 2005.
3. Alan Donovan, S.M. Thesis: *Converting Java programs to use generic libraries*, September 2004.
4. David Saff, S.M. Thesis: *Automated continuous testing to speed software development*, February 2004.
5. Stephen McCamant, S.M. Thesis: *Predicting problems caused by component upgrades*, January 2004.

M. Eng. Theses:

1. Jalali, Darioush, M.S. report: *Not all mutants are equally strong: Mutation operators should not be applied indiscriminately*, April 2014.
2. Coward, Forrest M.S. thesis, *Scout: Focusing Developers on Expressive Contracts*, March 2014.
3. Rigsby, Tyler, M.S. thesis: *Conflict Weighting in Verification Games*, December 2013.
4. Dietzel, Stephanie, M.S. thesis: *Improving and Extending Verigames*, June 2013.
5. Donohue, Kellen, M.S. thesis: *Celeriac: A Daikon .NET Front End*, June 2013.
6. Spishak, Eric, M.S. thesis: *Annotation File Format Specifictaion*, March 2013.
7. Abrahamson, Jenny, M.S. thesis: *Lightweight Visualizations of Distributed System Traces*, March 2013.
8. Rudd, Robert, M.Eng. thesis: *An improved scalable mixed-level approach to dynamic analysis of C and C++ programs*, January 2010.
9. Quinonez, Jaime, M.Eng. thesis: *Inference of Reference Immutability in Java*, May 2008 (also used for S.B. degree).
10. Papi, Matthew, M.Eng. thesis: *Practical Pluggable Types for Java*, May 2008 (also used for S.B. degree). Won the Charles and Jennifer Johnson Thesis Award.
11. Glasser, David, M.Eng. thesis: *Test factoring with* `amock`*: Generating readable unit tests from system tests*, August 2007 (also used for S.B. degree).

12. Xiao, Chen, M.Eng. thesis: *Performance enhancements for a dynamic invariant detector*, February 2007 (also used for S.B. degree).
13. Tschantz, Matthew, M.Eng. thesis: *Javari: Adding reference immutability to Java*, August 2006 (also used for S.B. degree). This research won the Anna Pogosyants undergraduate research prize. Won the Charles and Jennifer Johnson Thesis Award.
14. Guo, Philip, M.Eng. thesis: *A scalable mixed-level approach to dynamic analysis of C and C++ programs*, May 2006. Won the Charles and Jennifer Johnson Thesis Award.
15. Lin, Lee, M.Eng. thesis: *Improving adaptability via program steering*, August 2004.
16. Brun, Yuriy, M.Eng. thesis: *Software fault identification via dynamic analysis and machine learning*, August 2003.
17. Ne Win, Toh, M.Eng. thesis: *Theorem-proving distributed algorithms with dynamic analysis*, May 2003 (also used for S.B. degree). Won the Charles and Jennifer Johnson Thesis Award.
18. Birka, Adrian, M.Eng. thesis: *Compiler-enforced immutability for the Java language*, May 2003 (also used for S.B. degree).
19. Dodoo, Nii, M.Eng. thesis: *Selecting predicates for conditional invariant detection using cluster analysis*, September 2002 (also used for S.B. degree).
20. Morse, Ben, M.Eng. thesis: *A C/C++ front end for the Daikon dynamic invariant detection system*, August 2002 (also used for S.B. degree).
21. Rolfe, Alex, M.Eng. thesis: *Code versioning in a workflow management system*, May 2002 (also used for S.B. degree).
22. Nimmer, Jeremy W., M.Eng. thesis: *Automatic generation and checking of program specifications*, May 2002 (also used for S.B. degree). Won the Charles and Jennifer Johnson Thesis Award.
23. Harder, Michael, M.Eng. thesis: *Improving test suites via generated specifications*, May 2002 (also used for S.B. degree).
24. Dean, Laura G., M.Eng. thesis: *Improved simulation of Input/Output Automata*, September 2001 (also used for S.B. degree).

Current Masters Students

Not applicable.

Other significant student supervision

Doctoral theses, supervisor

1. Carlos Pacheco, *Directed random testing*, February 2009. Current employer: Google. Official chair was Daniel Jackson, but most of the work was done under my supervision.

Doctoral theses, reader

1. Samhi, Jordan, in progress.
2. Thayer, Kyle, in progress.
3. Goffi, Alberto, in progress, University of Verona.
4. Huang, Justin, in progress.

5. Nath, Aniruddh, in progress.
6. Bergan, Tom, *Avoiding State-Space Explosion in Multithreaded Programs with Input-Covering Schedules and Symbolic Execution*, June 2014.
7. Mohsen Vakilian, *Less is sometimes more in the automation of software evolution tasks*, June 2014, University of Illinois.
8. Lucia, Brandon, *System Support for Correctness and Performance Debugging of Shared-Memory Concurrent Programs*, June 2013.
9. Nikolić, Đurica, *A General Framework for Constraint-Based Static Analyses of Java Bytecode Programs*, April 2013, University of Verona.
10. Dietl, Werner, *Universe types: Topology, encapsulation, genericity, and tools*, July 2010, ETH Zurich.
11. Garbervetsky, Diego, *Parametric specification of dynamic memory utilization*, November 2007, University of Buenos Aires.
12. Tauber, Joshua A., *Verifiable compilation of I/O automata without global synchronization*, August 2004.
13. Sălcianu, Alexandru, *Pointer analysis for Java: Novel techniques and applications*, August 2006.
14. Demsky, Brian, *Data structure repair using goal-directed reasoning*, January 2006.
15. Marinov, Darko *Automatic Testing of Software with Structurally Complex Inputs*, December 2005.
16. Ajmani, Sameer, *Automatic software upgrades for distributed systems*, August 2004.
17. Raz, Orna, *Semantic anomaly detection in dynamic data feeds with incomplete specifications*, May 2004, Carnegie Mellon University.

Undergraduate projects and theses

1. Mackie, Christopher, B.S. thesis: *Preventing Signedness Errors in Numerical Computations in Java*, June 2013.
2. Mote, Nat, B.S. thesis: *Verification Games Type Systems*, December 2013.
3. Lam, Wing, B.S. thesis *When Tests Collide: The Impact of Order-Dependent Tests on Test Prioritization*, December 2013.
4. Sukkerd, Roykrong, B.S. thesis (2013): *Leveraging Data Invariants in Model Inference for Test Case Generation*, June 2013.
5. Vega, Timothy, Bachelor's project: *Traceur: Inferring Variable Control Flow Using Synoptic With Multiple Relation Types*, December 2012.
6. Snow, Gareth, Bachelor's project: *Analysis of Mutation Testing Tools*, 2010.
7. Ali, Mahmood, MIT Advanced Undergraduate Project: *IGJ type-checker*, August 2008.
8. Gebauer, Michael, MIT Advanced Undergraduate Project: *Interactive voting system for Groupthink specification exercise*, February 2006.
9. Iba, Aaron, MIT Advanced Undergraduate Project: *Robocraft execution engine*, May 2004.
10. Grall, Jonathan, MIT Advanced Undergraduate Project: *Robocraft execution engine*, May 2004.
11. Meghani, Samir, MIT Advanced Undergraduate Project: *Determining legal method call sequences in object interfaces*, May 2003.

Undergraduate research

Not including those in "undergraduate projects and theses", above.

1. Andrew Davies
2. Mark Davis
3. Donovan Hunt
4. Allen Liu
5. Nat Mote
6. Eric Spishak
7. Laure Thompson
8. Timothy Vega
9. Yoong Woo Kim
10. Jeff Gertler
11. Zachary Stein
12. Michael Sloan
13. Naomi Bancroft
14. Asumu Takikawa
15. Artem Melentyev
16. David Lazar
17. Peter Kalauskas
18. Tim Vega
19. David Koenig
20. Matt Mullen
21. John Marrero
22. Telmo Correa
23. Charles Tam
24. Stephie Wu
25. Jeff Yuan
26. Arjun Dayal
27. Sanjukta Pal
28. Eric Fellheimer
29. Pramook Khungurn
30. Kevin Chevalier
31. Kathryn Shih
32. Galen Pickard
33. Meng Mao
34. Joseph Sikoscow
35. Punyashloka Biswal
36. Jelani Nelson
37. Samir Meghani
38. Cemal Akcaba
39. Jeff Mellen
40. Vikash Mansinghka
41. Alan Dunn
42. Benjamin Wang
43. Deepali Garg
44. Emily Marcus
45. James Anderson

46. Stanley Cheung
47. Arjun Narayanswamy
48. Faisal Anwar
49. Gustavo Santos

Postdoctoral supervision

1. René Just
2. Werner Dietl
3. Yuriy Brun, now at University of Massachusetts
4. Danny Dig, now at Oregon State University
5. Yoav Zibin, now at Google
6. Sung Kim, now at HKUST

Research staff supervision

1. Benjamin Keller
2. Dan Brown
3. Doug Hoeger
4. David McArthur, now at Oracle
5. Mark Roberts
6. Suzanne Millstein
7. Jonathan Burke, now at Oracle
8. Timothy Pavlik, co-supervised with Zoran Popović
9. Gregory Sullivan, now at BAE AIT
10. Sandra Loosemore, now at CodeSourcery
11. Stephen J. Garland, now retired
12. Jeff Perkins, now at MIT CSAIL

RESEARCH ACTIVITIES

Funder	Title	Role	Amount (subcontracts)	Dates
DARPA	Verification Games	PI Co-PI, Zoran Popović	$4,237,384 ($291,603 Julia srl)	4/12 – 4/15
Oracle	Evaluation of type annotations and pluggable type-checking	PI	$50,000	5/12 – 4/13
DARPA	SPARTA: Static Program Analysis for Reliable, Trusted Apps	PI Co-PI, David Wetherall Co-PI, Tadayoshi Kohno	$4,915,037	6/2012 – 12/2015
Oracle	Evaluation of type annotations and pluggable type-checking	PI	$80,000	5/11 – 4/12

Funder	Title	Role	Amount (subcontracts)	Dates
DARPA	Crowdsourced Program Verification	PI Co-PI, Zoran Popović	$453,614	4/11 – 4/12
NSF	SHF: Medium: Combining Speculation with Continuous Validation for Software	PI Co-PI, David Notkin	$766,000	9/10 – 8/13
NSF	SHF: Small: Always-On Static and Dynamic Feedback	PI	$496,586	8/10 – 7/13
Microsoft	Speculation and Continuous Validation for Software Development	PI Co-PI, David Notkin	$25,000	6/10 – 5/11
ABB	Random test generation for programs	PI	$100,000	2010???
Google	Google Summer of Code	PI	$2,000	5/10 – 9/10
NSF	[ARRA] II-NEW: Practical Pluggable Type Systems	PI	$681,071	8/09 – 7/12
IBM	John Backus Award	PI	$150,000	10/09 – 09/12
Google	Practical pluggable types	PI	$75,000	5/09 – 4/10
IBM	Faculty Award	PI	$25,000	10/08 – 9/09
Toshiba Corp	Exploiting Dynamic and Static Tools for Improving Software Quality	Co-PI PI, Daniel Jackson	$153,000	1/08 – 3/09
Toshiba Corp		PI	$51K	1/08 – 3/08
Air Force Research Lab	Reincarnation of Streaming Application	Co-PI PI, Saman Amarasinghe Co-PI, Robert Miller	$250,000	10/07 – 12/08
DARPA	Program Analysis for Software and Security	PI	$462,363	6/07 – 5/09
Air Force Research Lab	Collaborative Learning for Security and Repair in Application Communities	PI Co-PI: Martin Rinard	$2,986,964 ($1,063,496 Determina Inc.)	7/06 – 6/08
	Invariant Detection for Program Productivity	PI	$40,000	
NSF	ITR	PI	$235,000	
Nokia, Inc.	Intelligent Composition of Heterogeneous Services	PI	$100,000	1/06 – 3/07
NSF	SoD-HCER: Testing Designs and Designing Tests	PI	$200,000	8/06 – 7/08
DARPA	Computer Science Study Panel (CS2P)	PI	$99,750	4/06 – 3/07

Funder	Title	Role	Amount (subcontracts)	Dates
AF Research Lab	Learn and Repair Tech for Self Healing Systems	PI Co-PI Martin Rinard	$983,314	7/04 – 12/05
NASA Ames Research Ctr	Improving Test Suites Via Operational Abstraction	PI	$80,000	3/04 – 9/06
NSF	ITR: Software Safety Mechanism for Medical Systems	Co-PI PI, Daniel Jackson Co-PI, Martin Rinard	$300,000	10/03 – 3/09
NSF	Improving Test Suites Via Generated Specifications	PI Co-PI Stephen Garland	$399,649	9/02 – 3/06
Deshpande MOU (MIT)	Automatically Generating Specifications	PI	$250,000	9/02 – 3/04
AT&T	???	PI	$50,000	???
TIBCO	???	PI	$13,200	6/02 – 9/02
NSF	CAREER: Automatically Generating Specifications to Improve Program Correctness and Maintainability	PI	$300,000	2/02 – 1/07
Raython Company	Automated Support for Reliable Software Evolution	PI	$50,000	9/01 – 8/02
Nippon Telegraph & Telephone	Dynamic Invariant Detection for Program Understanding and Reliability	PI	$200,000	7/01 – 6/04
Multi-sponsored Consortium	O2P Software Reliability for Oxygen: Determination and Validation of Communication Profiles	PI	$50,000	7/01 – 6/06

DOCUMENTATION OF TEACHING EFFECTIVENESS

3. Other Educational Contribution

At Rice University, in addition to teaching existing subjects, I introduced a new senior-level project-oriented laboratory in software engineering. I also re-designed from scratch the second subject in computing, which focuses on data structures and algorithms while also introducing C++ and principles for program design and structuring.

At MIT, I created a new graduate class (6.893, later 6.883) covering static program analysis, dynamic program analysis, and the synergy between them, with a particular emphasis on analyses that assist humans in performing programming tasks.

For 6.170 (Laboratory in Software Engineering), I introduced a new approach in which students revise and re-submit the assignments. This "re-turnin" gives students a chance to learn from and correct their mistakes. Students get a small taste of working on (their own) legacy code, helping them to appreciate the benefits of good design and documentation. Students are given a few re-turnin tries, but credit for the re-turnin is all-or-nothing, so students are motivated to apply the intellectual tools of the course: testing, reasoning, etc.

Also for 6.170, I helped create substantial infrastructure supporting turning in assignments, grading them (both for automated feedback to students and to assist graders), and other aspects. I created new assignments (now used nationwide as others have adopted them), updated the class by introducing many new program development tools, and developed new lectures on several topics. I addressed a serious and long-standing problem (that students were well aware of) by running plagiarism detection tools and prosecuting cheaters.

For the Undergraduate Practice Opportunities Program (UPOP), I created a popular segment on specifications. The Groupthink Specification Exercise uses a gameshow format in which students compete in defining the behavior of a telephone answering machine, and using their specifications to answer questions. I created, and distribute to other instructors, the content and supporting software; it has been used in multiple other institutions. I have also published two papers describing the activity.

I have been in charge of 6.187, a programming competition run each IAP, since its inception. This programming competition is unlike some, which emphasize speed, in that it uses much more complicated (and realistic) problem domains and runs for a full month, thus exercising and testing more realistic software development skills.

All my course materials are available on the web — sometimes excluding solutions and infrastructure, which are available on request.

SERVICE

Departmental service

1. Graduate admissions committee, 2016 – 2017
2. Executive committee, 2015 – 2016
3. BS/MS committee (chair), 2014
4. Graduate admissions committee, 2013 – 2014
5. ExCEL faculty recruiting, 2012 – 2013
6. Notkinfest (chair), Nov. 2012 – Feb. 2013
7. Best Senior Thesis award committee, 2011 – 2013
8. Graduate admissions committee, 2011 – 2012
9. BS/MS committee (chair), 2010, 2011
10. Curriculum revision committee, Jan. 2009 – June 2010
11. MIT EECS Graduate Admission Committee, Dec. 2006 – April 2007
12. MIT AA/EECS Faculty Search Committee, Jan. 2005 – June 2006
13. MIT EECS Sprowls/ACM Dissertation Award Committee, Fall 2004 – Spring 2007
14. MIT EECS Faculty advisor, Eta Kappa Nu (EECS honor society), Oct. 2002 – Nov 2008
15. MIT EECS Co-faculty advisor, ACM programming contest team, Spring 2001 – Spring 2001
16. MIT EECS Graduate Admission Committee, Dec. 2000 – April 2004
17. MIT EECS undergraduate counselor, Fall 2000 – Spring 2007

College service

1. MIT AA (Course 16) Faculty Search Committee, Fall 2000 – Spring 2001

University service

UW Extension, Advisory Board Member, Scala Certificate Program

Professional society and other service

1. Member, ACM SIGSOFT Outstanding Dissertation Award Committee, 2014
2. Member, Computing Research Association Education Committee (CRA-E), 2013 – 2015
3. Steering Committee member, PASTE (ACM Program Analysis for Software Tools and Engineering workshop), 2007 – 2008
4. Specification Lead, JCP-308 Annotations on Java Types, 2006 – present
5. Expert Group member, JCP-305 Annotations for Software Defect Detection, 2006 – present
6. Steering Committee chair, PASTE (ACM Program Analysis for Software Tools and Engineering workshop) , 2005 – 2007
7. Co-founded the WODA (Workshop on Dynamic Analysis) workshop series, which has run annually since 2003

Community service

Lots: founding non-profit organizations, serving on the boards of others, significant and sustained volunteering, etc. Some of this is even technology-focused. I don't see how any of this is relevant to my professional c.v., however.

International, national or governmental service

This is an incomplete list.

1. NSF review panel, 2012
2. NSF review panel, 2009
3. NSF review panel, 2006
4. Swiss National Science Foundation review panel, 2007
5. Swiss NSF review panel, 2006
6. NSF ITR review panel, 2004
7. Netherlands Organization for Scientific Research (NWO) review panel, 2004
8. NSF CAREER review panel, 2002
9. External reviewer, U.S. Congressional Office of Technology Assessment "Computer Software and Intellectual Property" project, 1991 – 1992