

Michael D. Ernst
Curriculum Vitae

Date: May 2008

Full Name: Michael D. Ernst

Department: University of Washington, Computer Science & Engineering

1. Date of Birth:

March 14, 1967

2. Citizenship:

USA

3. Education:

School	Degree	Date
MIT	S.B., EECS	June 1989
MIT	S.M., EECS	September 1992
U. of Washington	M.S., Computer Science & Engineering	March 1997
U. of Washington	Ph.D., Computer Science & Engineering	August 2000

4. Title of Thesis for Most Advanced Degree:

Dynamically Discovering Likely Program Invariants

5. Principal Fields of Interest:

Software engineering; programmer productivity tools and methodology; reverse engineering; program understanding; programming environments; program analysis; programming language design; formal methods; dynamic analysis; machine learning.

6. Employment:

Employer	Position	Beginning	Ending
Texas Instruments	Summer intern	May 1986	Sep. 1989
Microsoft	Software Design Engineer	Sep. 1992	Mar. 1993
Microsoft	Researcher	Mar. 1993	Aug. 1995
Rice University	Lecturer	Sept. 1995	May 1996
U. of Washington	Research Assistant	Sept. 1996	Aug. 2000
MIT	Assistant Professor	Sept. 2000	Feb. 2005
MIT	Associate Professor	Feb. 2005	Dec. 2008
U. of Washington	Associate Professor	Jan. 2009	present

7. Consulting Record:

Firm	Beginning	Ending
GraniteStream	Oct. 2001	May 2003
Mercury Interactive	July 2004	July 2005
Institute for Defense Analyses	Jan. 2006	present
Kestrel Technology	Nov. 2006	present
McKinsey & Co.	Sep. 2007	Jan. 2008
Sugarman & Sugarman	May 2008	present
Samsung	Dec 2009	present

8. University Committees, Other Assigned Duties:

Activity	Beginning	Ending
MIT EECS undergraduate counselor	Fall 2000	Spring 2007
MIT EECS Graduate Admission Committee	Dec. 2000	April 2004
MIT Course 16 Faculty Search Committee	Fall 2000	Spring 2001
MIT Co-faculty advisor, ACM programming contest team	Spring 2001	Spring 2001
MIT Faculty advisor, Eta Kappa Nu (EECS honor society)	Oct. 2002	Nov 2008
MIT EECS Sprowls/ACM Dissertation Award Committee	Fall 2004	Spring 2007
MIT AA/EECS Faculty Search Committee	Jan. 2005	June 2006
MIT EECS Graduate Admission Committee	Dec. 2006	April 2007
UW curriculum revision committee	Jan. 2009	present
UW chair, BS/MS committee	May 2010	present

9. Professional Service:

Activity	Beginning	Ending
External reviewer, U.S. Congressional Office of Technology Assessment “Computer Software and Intellectual Property” project	1991	1992
Steering Committee chair, PASTE (ACM Program Analysis for Software Tools and Engineering workshop)	2005	2007
Steering Committee member, PASTE (ACM Program Analysis for Software Tools and Engineering workshop)	2007	2008
Expert Group member, JCP-305 Annotations for Software Defect Detection	2006	present
Specification Lead, JCP-308 Annotations on Java Types	2006	present

In 2003, I co-founded the WODA (Workshop on Dynamic Analysis) workshop series, which has run annually since.

Program Committees

Activity	Year
Chair, IR 1995. ACM SIGPLAN Intermediate Representations Workshop	1995

RV 2001. Workshop on Runtime Verification	2001
NEPLS 4. Fourth New England Programming Languages and Systems Symposium	2001
SCAM 2001. IEEE International Workshop on Source Code Analysis and Manipulation	2001
Chair, NEPLS 5. Fifth New England Programming Languages and Systems Symposium	2002
NEPLS 6. Sixth New England Programming Languages and Systems Symposium	2002
RV 2002. Second Workshop on Runtime Verification	2002
FSE 2002. ACM Tenth International Symposium on the Foundations of Software Engineering	2002
PASTE 2002. ACM Workshop on Program Analysis for Software Tools and Engineering	2002
Co-chair, WODA 2003. Workshop on Dynamic Analysis	2003
eTX 2004. Eclipse Technology Exchange Workshop	2003
PL Day 2004. IBM Programming Language Day Workshop	2004
ICSE 2004. 26th International Conference on Software Engineering	2004
WODA 2004. Workshop on Dynamic Analysis	2004
PASTE 2004. ACM Workshop on Program Analysis for Software Tools and Engineering	2004
FTfJP 2004. Formal Techniques for Java-like Programs Workshop	2004
POPL 2005. ACM Symposium on Principles of Programming Languages	2005
ESDDT 2005 (“Bugs 2005”). Workshop on the Evaluation of Software Defect Detection Tools	2005
Co-chair, PASTE 2005. ACM Workshop on Program Analysis for Software Tools and Engineering	2005
VSTTE 2005. Verified Software: Theories, Tools, Experiments	2005
CC 2006. International Conference on Compiler Construction	2006
WODA 2006. Workshop on Dynamic Analysis	2006
PLDI 2006. ACM Conference on Programming Language Design and Implementation	2006
ISSTA 2006. International Symposium on Software Testing and Analysis	2006
ECOOP 2006. European Conference on Object-Oriented Programming	2006
eTX 2006. Eclipse Technology Exchange Workshop	2006
ICSE 2007 Education Track. 29th International Conference on Software Engineering	2007
TAP 2007. Tests and Proofs	2007
ESEC-FSE 2007. European Software Engineering Conference and ACM International Symposium on the Foundations of Software Engineering	2007
ISSTA 2008. International Symposium on Software Testing and Analysis	2008
HotSWUp 2008. ACM Workshop on Hot Topics in Software Upgrades	2008
ICSE 2009 Mentor Program. 31st International Conference on Software Engineering	2009
WODA 2009. Workshop on Dynamic Analysis	2009

In addition to program committee service noted above, I have served as an external referee for other conferences, as a referee for journals, and as a panelist for grant proposals. An incomplete list of that service is: 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), National Science Foundation (NSF), Netherlands Organization for Scientific Research (NWO), 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), Swiss National Science Foundation, 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).

10. Awards Received:

(Not including graduate fellowships or other graduate school and earlier awards.)

Award	Date
ICSE 1999 paper selected for expedited journal publication	May 1999
ICSE 2000 paper selected for expedited journal publication	June 2000
U. of Washington William Chan Memorial Dissertation Award	2000
Honorable mention, ACM doctoral dissertation competition	2000
ICSM 2001 Best dissertation of past three years	November 2001
NSF CAREER Award	February 2002
VMCAI 2003 paper selected for expedited journal publication	January 2003
IBM Eclipse Innovation Award	2003, 2004, 2005
ESEC/FSE 2003 ACM Distinguished Paper Award	September 2003
Ross Career Development Professorship of Software Technology (MIT)	July 2003
ICSE 2004 paper nominated for ACM Distinguished Paper Award	May 2004
ASE 2005 paper nominated for Best Paper Award	November 2005
IBM faculty award	June 2006
ISSTA 2006 paper selected for expedited journal publication	July 2006
ASE 2006 paper selected for expedited journal publication	September 2006
Most Innovative JSR of the Year (Sun Microsystems JCP Program)	May 2007
ICSE 2007 ACM Distinguished Paper Award	May 2007
ESEC/FSE 2007 ACM Distinguished Paper Award	September 2007
ASE 2007 paper selected for expedited journal publication	November 2007
ISSTA 2008 paper selected for expedited journal publication	July 2008
JavaOne 2009 Rock Star	May 2009
ISSTA 2009 ACM Distinguished Paper Award	July 2009
Inaugural IBM John Backus Award	August 2009

11. Organization Membership:

Organization

Association for Computing Machinery (ACM)
 Eta Kappa Nu (electrical engineering honor society)
 Sigma Xi (scientific research honor society)
 Tau Beta Pi (engineering honor society)

12. Patents and Patent Applications Pending:

1. Gideon A. Yuval and Michael Ernst, “Method and system for controlling unauthorized access to information distributed to users,” U.S. Patent 5,586,186, December 17, 1996.

13. Professional Registration:

None.

14. Major New Projects, Processes, Designs, or Systems:

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 <http://pag.csail.mit.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. Other researchers have since produced their own invariant detection systems. Almost 100 papers (that I know of; see <http://pag.csail.mit.edu/daikon/pubs-using/>) use Daikon as an integral part of their research methodology, and additional papers use Daikon as a test subject — for instance, to evaluate regression testing tools, because Daikon has both a version control repository and a test suite. 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. We have built sound, precise, and scalable systems that introduce parametric polymorphism (generic types) in Java programs. The built-in generics refactoring in Eclipse — the most widely-used Java integrated development environment (IDE) — stems from one part of this work.
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).
5. 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.
6. 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.

7. Eclat automatically generates test inputs. It uses a novel selection mechanism to present to the user only the candidate tests that are most likely to indicate bugs or deficiencies in an existing test suite. The Joe and Randoop systems implement feedback-directed random test generation, integrating test generation and execution to improve each one. Palulu implements model-directed random test generation, inferring and using temporal interface models to enable generation of complex but legal tests that are similar to, but not identical to, previously observed behavior. Given an initial test suite, the latter three systems propose additional tests that are likely to reveal errors. 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.
8. Javari is a language that extends Java by permitting the specification and enforcement of reference immutability constraints. A compiler for the language is available, and over 160,000 lines of Javari code exist.
9. We have proposed an extension to Java's annotation system that generalizes it and makes it useful for type qualifiers and other uses. Sun has agreed to incorporate the extended annotations (in Java parlance, "JSR 308") in the Java 7 language. We have built a reference implementation: a modification of the javac compiler that accepts the annotations and stores them in class files, and compile-time and load-time type checkers for type qualifiers, such as "non-null".
10. 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).
11. DynComp infers abstract types for Java, C, and C++ programs. Programmers often use primitive types such as `int` (integer) to stand for a variety of concepts: array index, character, day of month, file descriptor, seconds since 1970, etc. DynComp partitions the uses into finer-grained abstract types, one per concept. It is a dynamic analysis that tracks value flows through a program to determine which values interact with one another (and so can be assumed to be of related abstract types).
12. 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 in institutions from Boston to Seattle. I distribute, handouts, lecture slides, instructors' notes, and optional software that automates running the activity.
13. The PittSField tool, built in the Program Analysis group by Stephen McCamant, 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.
14. 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 much research has addressed making SAT compilation very fast. Today, solving problems by reducing them to SAT is a very common technique used in all fields of computer science.

15. 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.
16. 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.

Teaching Experience of Michael D. Ernst

1. Teaching Experience

Term	Subject	Title	Role	Type	Eval. survey
FT 95	Comp212	Intermediate Programming (Rice University)	Lectures, in charge	Lab	Yes
FT 95	Comp410	Software Engineering (Rice University)	Lectures, in charge	Design	Yes
FT 96	Comp210	Principles of Computing and Programming (Rice Univ.)	Recitations (2)	Lecture	Yes
ST 96	Comp212	Intermediate Programming (Rice University)	Lectures, in charge	Lab	Yes
FT 00	6.170	Laboratory in Software Engineering	Lectures, development	Lab	Yes
IAP 01	6.187	“6.370” Programming Competition	Lectures, in charge	Design	No
ST 01	6.170	Laboratory in Software Engineering	Lectures, in charge	Lab	Yes
ST 01	6.897	Modeling and Analyzing Complicated Systems	Some lectures	Lecture	Yes
FT 01	6.893	Program Analysis for Software Engineering	Lectures, in charge	Seminar	No
IAP 02	6.187	“6.370” Programming Competition	In charge	Design	No
IAP 02	6.EPW	Undergraduate Practice Opportunities Program	Some lectures	Lab	Yes
ST 02	6.033	Computer System Engineering	Recitations (2)	Lecture	Yes
FT 02	6.821	Programming Languages	Lectures, in charge	Lecture	Yes
IAP 03	6.187	“6.370” Programming Competition	In charge	Design	No
IAP 03	6.EPW	Undergraduate Practice Opportunities Program	Some lectures	Lab	Yes
ST 03	6.170	Laboratory in Software Engineering	Lectures, in charge	Lab	Yes
FT 03		Junior faculty leave			
IAP 04	6.187	“6.370” Programming Competition	In charge	Design	No
IAP 04	6.EPW	Undergraduate Practice Opportunities Program	Some lectures	Lab	Yes
ST 04	6.033	Computer System Engineering	Recitations (2)	Lecture	Yes
FT 04	6.821	Programming Languages	Lectures, in charge	Lecture	Yes
IAP 05	6.187	“6.370” Programming Competition	In charge	Design	No
IAP 05	6.EPW	Undergraduate Practice Opportunities Program	Some lectures	Lab	Yes
ST 05	6.170	Laboratory in Software Engineering	Lectures, in charge	Lab	Yes
FT 05	6.883	Program Analysis	Lectures, in charge	Lecture	Yes
IAP 06	6.187	“6.370” Programming Competition	In charge	Design	No
IAP 06	6.EPW	Undergraduate Practice Opportunities Program	Some lectures	Lab	Yes
ST 06	6.170	Laboratory in Software Engineering	Lectures, in charge	Lab	Yes
FT 06		Parental release			
IAP 07	6.187	“6.370” Programming Competition	In charge	Design	No
IAP 07	6.EPW	Undergraduate Practice Opportunities Program	Some lectures	Lab	Yes
ST 07	6.170	Laboratory in Software Engineering	Lectures, in charge	Lab	Yes
FT 07		Parental release			

2. Teaching Evaluation Data

(For courses taught during the last three academic years, that I was substantially involved in, and for which teaching evaluation data was collected.)

Teaching Experience of Michael D. Ernst

Term	Subject number	Total # students registered	Total # survey responses	Survey form used	Instructor teaching quality	Overall course quality
ST05	6.170	74	not rated (reviewer did not show up)			
FT05	6.883	20	16	EECS	6.0	5.6
ST06	6.170	96	56	EECS	5.8	5.7
FT06	Parental release					
ST07	6.170					
FT07	Parental release					

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.

Publications of Michael D. Ernst

Hyperlinks to all abstracts and papers are available at <http://pag.csail.mit.edu/~mernst/pubs/>.
(* indicates outgrowth of supervised student research.)

1. Books:

(None.)

2. Papers in Refereed Journals:

1. Ernst, Michael D. “Playing Konane Mathematically: A Combinatorial Game-Theoretic Analysis,” *UMAP Journal* vol. 16, no. 2, pp. 95–121, 1995.
2. Ernst, Michael D., Jake Cockrell, William G. Griswold, and David Notkin. “Dynamically Discovering Likely Program Invariants to Support Program Evolution,” *IEEE Transactions on Software Engineering*, vol. 27, pp. 1–25, February 2001.
3. Wilmer, Elizabeth L. and Michael D. Ernst. “Graphs induced by Gray codes,” *Discrete Mathematics*, vol. 257, pp. 585–598, November 28, 2002.
4. Ernst, Michael D., Greg J. Badros, and David Notkin. “An Empirical Analysis of C Preprocessor Use,” *IEEE Transactions on Software Engineering*, vol. 28, no. 12, pp. 1146–1170, December 2002.
5. Ne Win, Toh, Michael D. Ernst, Stephen J. Garland, Dilsun Kirli, and Nancy Lynch. “Using simulated execution in verifying distributed algorithms,” *Software Tools for Technology Transfer*, vol. 6, no. 1, pp. 67–76, July 2004. *
6. Burdy, Lilian, 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*, vol. 7, no. 3, pp. 212–232, June 2005. *
7. Ernst, Michael D., 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*, vol. 69, pp. 35–45, 2007. *

3. Papers in Proceedings of Refereed Conferences:

(Other than early versions of those above.)

1. Ernst, Michael D. and Bruce E. Flinchbaugh. “Image/map correspondence using curve matching,” *AAAI Robot Navigation Symposium*, 4 pp., March 1989.
2. Weise, Daniel, Roger F. Crew, Michael Ernst, and Bjarne Steensgaard. “Value dependence graphs: Representation without taxation,” *POPL 94: ACM Symposium on Principles of Programming Languages*, pp. 297–310, January 1994.

Publications of Michael D. Ernst

3. Ernst, Michael D., Todd D. Millstein, and Daniel S. Weld. “Automatic SAT-compilation of planning problems,” *IJCAI-97: 15th International Joint Conference on Artificial Intelligence*, Nagoya, Aichi, Japan, pp. 1169–1176, August 23–29, 1997.
4. Ernst, Michael D., Craig Kaplan, and Craig Chambers. “Predicate dispatching: A unified theory of dispatch,” *ECOOP 98: 12th European Conference on Object-Oriented Programming*, Brussels, Belgium, pp. 186–211, July 20–24, 1998.
5. Ernst, Michael D., Adam Czeisler, William G. Griswold, and David Notkin. “Quickly detecting relevant program invariants,” *ICSE 2000: 22nd International Conference on Software Engineering*, Limerick, Ireland, pp. 449–458, June 7–9, 2000.
6. Nimmer, Jeremy W. and Michael D. Ernst. “Static verification of dynamically detected program invariants: Integrating Daikon and ESC/Java,” *RV’01: First Workshop on Runtime Verification*, Paris, France, 22 pp., July 23, 2001. *
7. Kataoka, Yoshio, Michael D. Ernst, William G. Griswold, and David Notkin. “Automated support for program refactoring using invariants,” *ICSM’01, International Conference on Software Maintenance*, Florence, Italy, pp. 736–743, November 6–10, 2001.
8. Nimmer, Jeremy W. and Michael D. Ernst. “Automatic generation of program specifications,” *ISSTA’02, International Symposium on Software Testing and Analysis*, Rome, Italy, pp. 232–242, July 22–24, 2002. *
9. Nimmer, Jeremy W. and Michael D. Ernst. “Invariant inference for static checking: An empirical evaluation,” *FSE’02, 10th International Symposium on the Foundations of Software Engineering*, Charleston, SC, pp. 11–20, November 18–22, 2002. *
10. Harder, Michael, Jeff Mellen, and Michael D. Ernst. “Improving test suites via operational abstraction,” *ICSE’03, 25th International Conference on Software Engineering*, Portland, Oregon, pp. 60–71, May 6–8, 2003. *
11. Ernst, Michael D. “Static and dynamic analysis: Synergy and duality,” *WODA 2003: ICSE Workshop on Dynamic Analysis*, Portland, Oregon, pp. 24–27, May 9, 2003.
12. McCamant, Stephen and Michael D. Ernst. “Predicting problems caused by component upgrades,” *ESEC/FSE 2003: 10th European Software Engineering Conference and the 11th ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Helsinki, Finland, pp. 287–296, September 3–5, 2003. *
13. Saff, David and Michael D. Ernst. “Reducing wasted development time via continuous testing,” *ISSRE 2003, Fourteenth International Symposium on Software Reliability Engineering*, Denver, CO, pp. 281–292, November 17–20, 2003. *
14. Lin, Lee and Michael D. Ernst. “Improving reliability and adaptability via program steering,” *ISSRE 2003, Fourteenth International Symposium on Software Reliability Engineering*, Denver, CO, pp. 313–314, November 17–20, 2003. *

Publications of Michael D. Ernst

15. Saff, David and Michael D. Ernst. “Continuous testing in Eclipse,” *eTX, 2nd Eclipse Technology Exchange Workshop*, Barcelona, Spain, 15 pp., March 30, 2004. *
16. Brun, Yuriy and Michael D. Ernst. “Finding latent code errors via machine learning over program executions,” *ICSE 2004, 25th International Conference on Software Engineering*, Edinburgh, Scotland, pp. 480–490, May 26–28, 2004. *
17. Saff, David and Michael D. Ernst. “Automatic mock object creation for test factoring,” *PASTE’04, ACM SIGPLAN/SIGSOFT Workshop on Program Analysis for Software Tools and Engineering*, Washington, DC, USA, pp. 49–51, June 7–8, 2004. *
18. McCamant, Stephen and Michael D. Ernst. “Early identification of incompatibilities in multi-component upgrades,” *ECOOP 2004, Object-Oriented Programming, 18th European Conference*, Oslo, Norway, pp. 440–464, June 14–18, 2004. *
19. Saff, David and Michael D. Ernst. “An experimental evaluation of continuous testing during development,” *ISSTA 2004, International Symposium on Software Testing and Analysis*, Boston, MA, USA, pp. 76–85, July 12–14, 2004. *
20. Lin, Lee and Michael D. Ernst. “Improving adaptability via program steering,” *ISSTA 2004, International Symposium on Software Testing and Analysis*, Boston, MA, USA, pp. 206–216, July 12–14, 2004. *
21. Donovan, Alan, Adam Kieżun, Matthew S. Tschantz, and Michael D. Ernst. “Converting Java programs to use generic libraries,” *OOPSLA 2004, Object-Oriented Programming Systems, Languages, and Applications*, Vancouver, BC, Canada, pp. 15–34, October 26–28, 2004. *
22. Birka, Adrian and Michael D. Ernst. “A practical type system and language for reference immutability,” *OOPSLA 2004, Object-Oriented Programming Systems, Languages, and Applications*, Vancouver, BC, Canada, pp. 35–49, October 26–28, 2004. *
23. Perkins, Jeff H. and Michael D. Ernst. “Efficient incremental algorithms for dynamic detection of likely invariants,” *FSE 2004, ACM SIGSOFT 12th Symposium on the Foundations of Software Engineering*, Newport Beach, CA, USA, pp. 23–32, November 2–4, 2004. *
24. Pacheco, Carlos and Michael D. Ernst. “Eclat: Automatic generation and classification of test inputs,” *ECOOP 2005, Object-Oriented Programming, 19th European Conference*, Glasgow, Scotland, pp. 504–527, July 25–29, 2005. *
25. Williams, Amy, William Thies, and Michael D. Ernst. “Static deadlock detection for Java libraries,” *ECOOP 2005, Object-Oriented Programming, 19th European Conference*, Glasgow, Scotland, pp. 602–629, July 25–29, 2005. *
26. Ernst, Michael D. “Verification for legacy programs,” *VSTTE 2005, Verified Tools: Theories, Tools, Experiments*, Zürich, Switzerland, 6 pp., October 10–13, 2005.

Publications of Michael D. Ernst

27. Tschantz, Matthew S. and Michael D. Ernst. “Javari: Adding reference immutability to Java,” *OOPSLA 2005, Object-Oriented Programming Systems, Languages, and Applications*, San Diego, CA, USA, pp. 211–230, October 18–20, 2005. *
28. Artzi, Shay and Michael D. Ernst. “Using predicate fields in a highly flexible industrial control system,” *OOPSLA 2005, Object-Oriented Programming Systems, Languages, and Applications*, San Diego, CA, USA, pp. 319–330, October 18–20, 2005. *
29. Saff, David, Shay Artzi, Jeff H. Perkins, and Michael D. Ernst. “Automatic test factoring for Java,” *ASE 2005: 21st Annual International Conference on Automated Software Engineering*, Long Beach, CA, USA, pp. 114–123, November 9–11, 2005. *
30. Ernst, Michael D., Raimondas Lencevicius, and Jeff H. Perkins. “Detection of web service substitutability and composability,” *WS-MaTe 2006, International Workshop on Web Services — Modeling and Testing*, Palermo, Italy, pp. 123–135, June 9, 2006.
31. Demsky, Brian, Michael D. Ernst, Philip J. Guo, Stephen McCamant, Jeff H. Perkins, and Martin Rinard. “Inference and enforcement of data structure consistency specifications,” *ISSTA 2006, International Symposium on Software Testing and Analysis*, Portland, ME, USA, pp. 233–243, July 18-20, 2006. *
32. Guo, Philip J., Jeff H. Perkins, Stephen McCamant, and Michael D. Ernst. “Dynamic inference of abstract types,” *ISSTA 2006, International Symposium on Software Testing and Analysis*, Portland, ME, USA, pp. 255–265, July 18-20, 2006. *
33. d’Amorim, Marcelo, Carlos Pacheco, Darko Marinov, Tao Xie, and Michael D. Ernst. “An empirical comparison of automated generation and classification techniques for object-oriented unit testing,” *ASE 2006: 21st Annual International Conference on Automated Software Engineering*, Tokyo, Japan, pp. 59–68, September 20–22, 2006. *
34. Artzi, Shay, Michael D. Ernst, David Glasser, Adam Kiezun, Carlos Pacheco, and Jeff H. Perkins. “Finding the needles in the haystack: Generating legal test inputs for object-oriented programs,” *M-TOOS 06, 1st Workshop on Model-Based Testing and Object-Oriented Systems*, Portland, OR, USA, 8 pp., October 23, 2006. *
35. Ernst, Michael D. “The Groupthink specification exercise,” in *Software Engineering Education in the Modern Age: Challenges and Possibilities*, vol. 4309 of LNCS, pp. 89–107, December, 2006.
36. Kiezun, Adam, Michael D. Ernst, Frank Tip, and Robert M. Fuhrer. “Refactoring for parameterizing Java classes,” *ICSE 2007, 29th International Conference on Software Engineering*, Minneapolis, MN, USA, pp. 437–446, May 23–25, 2007. *
37. Pacheco, Carlos, Shuvendu K. Lahiri, Michael D. Ernst, Thomas Ball. “Feedback-directed random test generation,” *ICSE 2007, 29th International Conference on Software Engineering*, Minneapolis, MN, USA, pp. 75–84, May 23–25, 2007. *

Publications of Michael D. Ernst

38. Stephen McCamant and Michael D. Ernst. “A simulation-based proof technique for dynamic information flow,” *PLAS 2007: ACM SIGPLAN Workshop on Programming Languages and Analysis for Security*, San Diego, California, USA, June 14, 2007. *
39. Sunghun Kim and Michael D. Ernst. “Which warnings should I fix first?” *ESEC/FSE 2007: 11th European Software Engineering Conference and the 15th ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Dubrovnik, Croatia, pp. 45–54, September 5–7, 2007.
40. Yoav Zibin, Alex Potanin, Mahmood Ali, Shay Artzi, Adam Kieun, and Michael D. Ernst. “Object and reference immutability using Java generics,” *ESEC/FSE 2007: 11th European Software Engineering Conference and the 15th ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Dubrovnik, Croatia, pp. 75–84, September 5–7, 2007. *
41. Artzi, Shay, Adam Kiezun, David Glasser, and Michael D. Ernst. “Combined static and dynamic mutability analysis,” *ASE 2007: 22nd International Conference on Automated Software Engineering*, Atlanta, GA, USA, pp. 104–113, November 7–9, 2007. *
42. McCamant, Stephen and Michael D. Ernst. “Quantitative Information Flow as Network Flow Capacity,” *Proceedings of the ACM SIGPLAN 2008 Conference on Programming Language Design and Implementation*, Tucson, AZ, USA, June 9–11, 2008. *
- Quinonez, Jaime, Matthew S. Tschantz, and Michael D. Ernst. “Inference of reference immutability,” *ECOOP 2008 Object-Oriented Programming, 22nd European Conference*, Paphos, Cyprus, July 9–11, 2008. *
43. Artzi, Shay, Sunghun Kim, and Michael D. Ernst. “ReCrash: Making software failures reproducible by preserving object states,” *ECOOP 2008 Object-Oriented Programming, 22nd European Conference* Paphos, Cyprus, July 9–11, 2008. *
44. Papi, Matthew M., Mahmood Ali, Telmo Luis Correa Jr., Jeff H. Perkins, and Michael D. Ernst. “Practical pluggable types for Java,” *ISSTA 2008, Proceedings of the 2008 International Symposium on Software Testing and Analysis* Seattle, WA, USA, July 22–24, 2008. *
45. Artzi, Shay, Adam Kiezun, Julian Dolby, Frank Tip, Danny Dig, Amit Paradkar, and Michael D. Ernst. “Finding bugs in dynamic web applications,” *ISSTA 2008, Proceedings of the 2008 International Symposium on Software Testing and Analysis*, Seattle, WA, USA, July 22–24, 2008. *

4. Other Major Publications:

(Other than early versions of those above.)

1. Ernst, Michael D. “Adequate Models for Recursive Program Schemes,” Bachelor’s thesis, MIT Department of Electrical Engineering and Computer Science, 42 pp., June 1989.
2. Ernst, Michael D. (editor). “Intellectual property in Computing: (How) should software be protected? An industry perspective,” MIT Artificial Intelligence Laboratory Memo 1369, 26 pp., May 1992. Video published as MIT Artificial Intelligence Laboratory Video 7, October 1990.

Publications of Michael D. Ernst

3. Ernst, Michael D. “Serializing Parallel Programs by Removing Redundant Computation,” Master’s Thesis, MIT Department of Electrical Engineering and Computer Science, 95 pp., September 1992.
4. Ernst, Michael D. (editor). *Proceedings of IR ’95: Intermediate Representations Workshop*, January 22, 1995. ACM SIGPLAN Notices 30(3), 128 pp., March 1995.
5. Ernst, Michael D. (editor). “Dynamically Detecting Likely Program Invariants,” Ph.D. dissertation, University of Washington, Department of Computer Science & Engineering, 142 pp., August 2000.
6. Notkin, David, Marc Donner, Michael D. Ernst, Michael Gorlick, and E. James Whitehead, Jr., “Panel: Perspectives on software engineering” (position statement), *ICSE 2002, 24th International Conference on Software Engineering*, Montreal, Canada, pp. 699–702, May 16–18, 2002.
7. Cook, Jonathan E. and Michael D. Ernst (editors). *Proceedings of WODA 2003: ICSE Workshop on Dynamic Analysis*, 52 pp., May 9, 2003.
8. Ernst, Michael D. and Thomas Jensen (editors). *Proceedings of PASTE 2005, ACM SIGPLAN/SIGSOFT Workshop on Program Analysis for Software Tools and Engineering*, 119 pp., September 5–6, 2005.
9. Ernst, Michael D. and Jeff H. Perkins. “Learning from Executions: Dynamic Analysis for Software Engineering and Program Understanding,” Tutorial at *ASE 2005: 20th Annual International Conference on Automated Software Engineering*, Long Beach, CA, USA, November 7, 2005. *
10. Ernst, Michael D. “Annotations on Java types: JSR 308 working document,” 35 pp., November 12, 2007. *

5. Internal Memoranda and Progress Reports:

(Other than early versions of those above.)

1. Ernst, Michael D. and Gideon Yuval. “Heraclitean encryption,” Microsoft Research technical report MSR-TR-94-13, March 3, 1994.
2. Ernst, Michael D. “Practical fine-grained static slicing of optimized code,” Microsoft Research technical report MSR-TR-94-14, July 26, 1994.
3. Ernst, Michael D. “Slicing pointers and procedures (abstract),” Microsoft Research technical report MSR-TR-95-23, January 13, 1995.
4. Ernst, Michael D., William G. Griswold, Yoshio Kataoka, and David Notkin. “Dynamically discovering pointer-based program invariants,” University of Washington Department of Computer Science and Engineering technical report UW-CSE-99-11-02, November 16, 1999. Revised March 17, 2000.
5. Ne Win, Toh and Michael D. Ernst. “Verifying distributed algorithms via dynamic analysis and theorem proving,” MIT Laboratory for Computer Science technical report 841, May 25, 2002. *

Publications of Michael D. Ernst

6. Dodoo, Nii, Lee Lin, and Michael D. Ernst. “Selecting, refining, and evaluating predicates for program analysis,” MIT Laboratory for Computer Science technical report MIT-LCS-TR-914, July 21, 2003. *
7. Saff, David, Marat Boshernitsan, and Michael D. Ernst. “Theories in Practice: Easy-to-write specifications that catch bugs,” by MIT Computer Science and Artificial Intelligence Laboratory technical report MIT-CSAIL-TR-2008-002, January 14, 2008. *

6. Invited Lectures:

(Only talks since 2000 are listed. Talks at closed events (e.g., PI meetings) are not listed. Talks given by co-authors are not listed.)

“Dynamically Detecting Likely Program Invariants”

Cornell University, February 15, 2000

University of Colorado at Boulder, February 24, 2000

University of California at San Diego, February 29, 2000

University of California at Berkeley, March 2, 2000

Stanford University, March 6, 2000

Georgia Institute of Technology, March 23, 2000

University of Massachusetts at Amherst, March 29, 2000

Carnegie Mellon University, April 3, 2000

University of Maryland, April 5, 2000

Massachusetts Institute of Technology, April 10, 2000

Boston University, September 25, 2000

University of Virginia, “Top Gun” Distinguished Lecture Series, November 13, 2000

Tufts University, November 20, 2000

Brown University, February 1, 2001

University of Southern California, February 21, 2001

Compaq Systems Research Center, February 22, 2001

University of California at Irvine, February 23, 2001

International Conference on Software Maintenance (ICSM), Florence, Italy, November 8, 2001

“Playing Konane Mathematically with Combinatorial Game Theory”

MIT, January 17, 2001

“Refactoring and static verification: Two applications of dynamic invariant detection”

University of Wisconsin at Madison, May 14, 2001

IBM T.J. Watson Research Center, August 8, 2001

Williams College, November 30, 2001

University of Washington, January 10, 2002

Rice University, January 15, 2002

McMaster University, January 25, 2002

“The future of software engineering” (panel)

International Conference on Software Engineering (ICSE), Montreal, Canada, May 18, 2001

Publications of Michael D. Ernst

“Static verification of dynamically detected program invariants”

Workshop on Runtime Verification (RV’01), Paris, France, July 23, 2001

Java Verification Workshop (JVW’01), Portland, Oregon, January 13, 2002

“Overview of dynamic invariant detection”

Brunel University, July 24, 2001

“Automated Support for Program Refactoring using Invariants”

International Conference on Software Maintenance (ICSM), Florence, Italy, November 9, 2001

“Using dynamic analysis to assist program verification”

Cambridge University, England, July 25, 2002

“Improving test suites via operational abstraction”

Carnegie Mellon University, February 3, 2003

Microsoft Research, May 2, 2003

International Conference on Software Engineering (ICSE), Portland, Oregon, May 6, 2003

IBM T.J. Watson Research Center (Distinguished Lecture), June 16, 2003

“Predicting problems caused by component upgrades”

University of Pittsburgh, February 4, 2003

Cambridge University and Microsoft Research, December 10, 2003

“Static and dynamic analysis: Synergy and duality”

ICSE Workshop on Dynamic Analysis (WODA), Portland, Oregon, May 9, 2003

Dagstuhl workshop Understanding Program Dynamics (keynote), December 1, 2003

Program Analysis and Software Tools for Engineering (PASTE) workshop (keynote), June 7, 2004

“Comparing dynamic program behaviors”

Dagstuhl workshop Understanding Program Dynamics, December 2, 2003

“Using dynamic analysis to assist theorem proving”

University of Limerick, December 12, 2003

“Improving adaptability via program steering”

Carnegie Mellon University, April 27, 2004

“Making the most of impoverished test suites: Automatic classification of test inputs and test factoring”

UW/MSR Summer Institute on Trends in Testing: Theory, Techniques and Tools (keynote), August 24, 2004

Philips Medical, October 4, 2004

ABB, November 19, 2004

“A practical type system and language for reference immutability”

Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA), October 26, 2004

“Integrating reference immutability into the Java mainstream”

OOPSLA Workshop on the Java Platform: Tiger and Beyond, October 28, 2004

“Learning and repair techniques for self-healing systems”

DARPA Self-Regenerative Systems meeting, July 12, 2005

Publications of Michael D. Ernst

- “Verification for legacy programs”
Conference on Verified Software: Theories, Tools, Experiments, October 10, 2005
- “Eclat: Automatic generation and classification of test inputs”
University of Arizona, October 27, 2005
- “Learning from executions: Dynamic analysis for program understanding and software engineering”
Tutorial at Conference on Automated Software Engineering 2005, November 7, 2005
- “Javari: Adding reference immutability to Java”
University of California at San Diego, November 8, 2005
Harvard University, February 22, 2006
- “Evaluating systematic and random testing”
Indian Institute of Technology, Delhi, September 28, 2006
Microsoft Research India, October 10, 2006
- “Choosing the right tests: test selection via operational abstraction”
IBM India Research Lab, September 28, 2006
- “Combined static and dynamic mutability analysis”
Tata Research Development & Design Centre, September 29, 2006
IBM T.J. Watson Research Center, November 28, 2006
University of California at Berkeley, December 11, 2006
- “Dynamic inference of abstract types”
Microsoft Research, December 12, 2006
- “Refactoring for parameterizing Java classes”
Stanford University, January 8, 2007
Harvard University, March 7, 2007
- “User-defined type systems for error detection and prevention”
QCon, November 9, 2007
- “Self-defending software: Collaborative learning for security”
University of Washington, April 1, 2008
- “Preventing bugs with pluggable type-checking for Java”
J-Spring, April 16, 2008
Javoxx, Dec 2008
- “Scalable pluggable types”
Dagstuhl workshop Scalable Program Analysis, April 14–18, 2008
- “Upcoming Java programming-language changes”
JavaOne, May 6, 2008 (with Alex Buckley)
- “Inference of reference immutability”
University of Saarland, July 4, 2008 ECOOP 2008, July 11, 2008

Publications of Michael D. Ernst

“ReCrash: Making software failures reproducible by preserving object states”

University of Saarland, July 8, 2008

ECOOP 2008, July 11, 2008

“Building and using pluggable type systems with the Checker Framework”

ECOOP 2008, July 10, 2008

“Practical pluggable types for Java”

Max Planck Institute for Software Systems, July 17, 2008

ISSTA 2008, July 23, 2008

Theses Supervised by Michael D. Ernst

Summary

	Total	Completed	In Progress
S.B.	5	5	0
S.M.	5	5	0
M.Eng.	18	16	2
Doctoral			
As Supervisor:	4	1	3
As Reader:	7	6	1

S. B. Theses:

(Not including those listed under M.Eng. theses, below. Includes Advanced Undergraduate Projects that involve significant faculty supervision.)

Meghani, Samir, *Determining legal method call sequences in object interfaces*, AUP, May 2003.

Grall, Jonathan, *Robocraft execution engine*, AUP, May 2004.

Iba, Aaron, *Robocraft execution engine*, AUP, May 2004.

Gebauer, Michael, *Interactive voting system for Groupthink specification exercise*, AUP, February 2006.

Ali, Mahmood, *IGJ type-checker*, August 2008.

S. M. Theses:

McCamant, Stephen, *Predicting problems caused by component upgrades*, January 2004.

Saff, David, *Automated continuous testing to speed software development*, February 2004.

Donovan, Alan, *Converting Java programs to use generic libraries*, September 2004.

Williams, Amy, *Static detection of deadlock for Java libraries*, May 2005.

Pacheco, Carlos, *Eclat: Automatic generation and classification of test inputs*, June 2005.

M. Eng. Theses:

Dean, Laura G., *Improved simulation of Input/Output Automata*, September 2001 (also used for S.B. degree).

Harder, Michael, *Improving test suites via generated specifications*, May 2002 (also used for S.B. degree).

Nimmer, Jeremy W., *Automatic generation and checking of program specifications*, May 2002 (also used for S.B. degree). Won the Charles and Jennifer Johnson Thesis Award.

Rolfe, Alex, *Code versioning in a workflow management system*, May 2002 (also used for S.B. degree).

Morse, Ben, *A C/C++ front end for the Daikon dynamic invariant detection system*, August 2002 (also used for S.B. degree).

Theses Supervised by Michael D. Ernst

Dodoo, Nii, *Selecting predicates for conditional invariant detection using cluster analysis*, September 2002 (also used for S.B. degree).

Birka, Adrian, *Compiler-enforced immutability for the Java language*, May 2003 (also used for S.B. degree).

Ne Win, Toh, *Theorem-proving distributed algorithms with dynamic analysis*, May 2003 (also used for S.B. degree). Won the Charles and Jennifer Johnson Thesis Award.

Brun, Yuriy, *Software fault identification via dynamic analysis and machine learning*, August 2003.

Lin, Lee, *Improving adaptability via program steering*, August 2004.

Guo, Philip, *A scalable mixed-level approach to dynamic analysis of C and C++ programs*, May 2006. Won the Charles and Jennifer Johnson Thesis Award.

Tschantz, Matthew, *Javari: Adding reference immutability to Java*, August 2006 (also used for S.B. degree). This research won the Anna Pogogyants undergraduate research prize. Won the Charles and Jennifer Johnson Thesis Award.

Xiao, Chen, *Performance enhancements for a dynamic invariant detector*, February 2007 (also used for S.B. degree).

Glasser, David, *Test factoring with amock: Generating readable unit tests from system tests*, August 2007 (also used for S.B. degree).

Papi, Matthew, *Practical Pluggable Types for Java*, May 2008 (also used for S.B. degree). Won the Charles and Jennifer Johnson Thesis Award.

Quinonez, Jaime, *Inference of Reference Immutability in Java*, May 2008 (also used for S.B. degree).

Rudd, Robert, expected August 2009.

Ali, Mahmood, *Type inference tool for IGJ*, expected August 2009.

Doctoral Theses, Supervisor:

(Not including students working toward a Master's degree whose terminal degree is the PhD.)

McCamant, Stephen, *Quantitative information-flow tracking for real systems*, May 2008.

Pacheco, Carlos, *Directed random testing*, February 2009.

Artzi, Shay, *Model-based testing*, in progress (expected June 2009).

Kiežun, Adam, *Random and exhaustive testing*, in progress (expected June 2009).

Doctoral Theses, Reader:

Raz, Orna, *Semantic anomaly detection in dynamic data feeds with incomplete specifications*, May 2004, Carnegie Mellon University.

Ajmani, Sameer, *Automatic software upgrades for distributed systems*, August 2004.

Tauber, Joshua A., *Verifiable compilation of I/O automata without global synchronization*, August 2004.

Theses Supervised by Michael D. Ernst

Marinov, Darko *Automatic Testing of Software with Structurally Complex Inputs*, December 2005.

Demsky, Brian, *Data structure repair using goal-directed reasoning*, January 2006.

Sălcianu, Alexandru, *Pointer analysis for Java: Novel techniques and applications*, August 2006.

Garbervetsky, Diego, *Parametric specification of dynamic memory utilization*, November 2007, University of Buenos Aires.

Dietl, Werner, in progress (expected spring 2009), ETH Zurich.

Postdoctoral Associates and Fellows Supervised by Michael D. Ernst

Current Postdocs

(none)

Previous Postdocs

Name	Title	Current employer	Current Position
Yoav Zibin	Oct. 2006 – Aug. 2007	Come2Play Corporation	CTO
Sung Kim	Sep. 2006 – July 2008	Hong Kong U. of Sci. & Tech.	Assistant Professor
Danny Dig	Nov. 2007 – present	U. of Illinois	Postdoctoral Associate