Jeremy G. Siek

Indiana University
School of Informatics and Computing
Division of Computer Science
Lindley Hall
150 S. Woodlawn Ave.
Bloomington, IN 47405
United States of America

Email: jsiek@indiana.edu
Website: http://homes.soic.indiana.edu/∼jsiek

Research Interests

Research in programming language design and implementation to improve support for software libraries and domain specific languages, including research in type systems, program generation, and performance optimization.

Education

2005 **Indiana University**
Ph.D. Computer Science, *A Language for Generic Programming*
Advisor: Andrew Lumsdaine

1999 **University of Notre Dame**
M.S. Computer Science and Engineering
A Modern Framework for Portable High Performance Numerical Linear Algebra
Advisor: Andrew Lumsdaine

1997 **University of Notre Dame**
B.S. Mathematics

Professional Research Experience

2013–present **Indiana University**, Bloomington, IN
Associate Professor

2007–2013 **University of Colorado**, Boulder, CO
Assistant Professor

2006–2007 **University of Colorado**, Boulder, CO
Visiting Assistant Professor

2006–2007 **LogicBlox**, Atlanta, GA
Research Scientist

2005–2006 **Rice University**, Houston, TX
Post-doctoral Research Associate

2000–2005 **Indiana University**, Bloomington, IN
Research Assistant

Summer 2001 **AT&T Labs–Research**, Florham Park, NJ
Summer Manager

1999–2000 **Silicon Graphics Inc.**, Mountain View, CA
Intern

Honors

2010 **Distinguished Visiting Fellowship**
Scottish Informatics & Computer Science Alliance (SICSA) June 1 - July 31

2009-2014 **NSF Faculty Early Career Development (CAREER) Award**
Bridging the Gap Between Prototyping and Production

Impact

The TypeScript dialect of JavaScript, designed by Anders Hejlsberg and his team at Microsoft, uses ideas from my papers on gradual typing: *Gradual Typing for Functional Languages* and *Gradual Typing for Objects*.

The Implicits feature of the Scala programming language, designed by Martin Odersky and his colleagues, uses ideas from my paper *Essential Language Support for Generic Programming*.

My Ph.D. thesis laid the foundation for the Concepts feature that was proposed for C++2011. While that proposal did not become part of the standard, work is continuing towards making it part of a future version of the C++ standard.

Publications

Books

Jeremy G. Siek, Lee-Quan Lee, and Andrew Lumsdaine. *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley, 2002.

Book Chapters

Jeremy G. Siek and Andrew Lumsdaine. *Advances in Software Tools for Scientific Computing*, chapter A Modern Framework for Portable High Performance Numerical Linear Algebra. Springer, 2000.

Journal Articles

Thomas Nelson, Geoffrey Belter, Jeremy G. Siek, Elizabeth Jessup, and Boyana Norris. Reliable generation of high-performance matrix algebra. *ACM Transactions on Mathematical Software (TOMS)*, 2012. Submitted for review.

Erik Silkensen and Jeremy G. Siek. Well-typed islands parse faster. *Lecture Notes in Computer Science*, Volume 7829, 2012. Revised and Selected Papers from the Trends in Functional Programming 13th International Symposium, June 12-14, 2012.

David Broman and Jeremy G. Siek. Modelyze: a gradually typed host language for embedding equation-based modeling languages. *Science of Computer Programming*, 2012. submitted for review, 58 pages.

Jeremy G. Siek. The C++0x "concepts" effort. *Lecture Notes in Computer Science*, 7470:175–216, 2012. Special Issue Generic and Indexed Programming.

Daniel P. Friedman, Abdulaziz Ghuloum, Jeremy G. Siek, and Onnie Lynn Winebarger. Improving the lazy krivine machine. *Higher Order and Symbolic Computation*, 20(3):271–293, 2007.

Jeremy G. Siek and Andrew Lumsdaine. A language for generic programming in the large. *Science of Computer Programming*, 76:423–465, September 2011.

Ronald Garcia, Jaakko Järvi, Andrew Lumsdaine, Jeremy G. Siek, and Jeremiah Willcock. An extended comparative study of language support for generic programming. *Journal of Functional Programming*, 17(2):145–205, March 2007.

Ian Karlin, Elizabeth Jessup, Geoffrey Belter, and Jeremy G. Siek. Parallel memory prediction for fused linear algebra kernels. *SIGMETRICS Perform. Eval. Rev.*, 38:43–49, March 2011.

Conference Papers

Justin Gottschlich, Maurice Herlihy, Gilles Pokam, and Jeremy G. Siek. Visualizing transactional memory. In *International Conference on Parallel Architectures and Compilation Techniques (PACT)*, September 2012. 19% acceptance rate.

Devin Coughlin, Bor-Yuh Evan Chang, Amer Diwan, and Jeremy G. Siek. Measuring enforcement windows with symbolic trace interpretation. In *International Symposium of Software Testing and Analysis*, July 2012. 29% acceptance rate.

Wieyu Miao and Jeremy G. Siek. Pattern-based traits. In *Symposium on Applied Computing*, March 2012. 26% acceptance rate, 8 pages.

Amal Ahmed, Robert Bruce Findler, Jeremy G. Siek, and Philip Wadler. Blame for All. In *ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, January 2011. 23% acceptance rate.

Weiyu Miao and Jeremy G. Siek. Incremental type-checking for type-reflective metaprograms. In *GPCE '10: Proceedings of the International Conference on Generative Programming and Component Engineering*, 2010. 30% acceptance rate.

Justin E. Gottschlich, Manish Vachharajani, and Jeremy G. Siek. An efficient software transactional memory using commit-time invalidation. In *Proceedings of the 8th annual IEEE/ACM International Symposium on Code Generation and Optimization*, CGO '10, pages 101–110, New York, NY, USA, 2010. ACM. 41% acceptance rate.

Jeremy G. Siek and Philip Wadler. Threesomes, with and without blame. In *POPL '10: Proceedings of the 37th annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 365–376, New York, NY, USA, 2010. ACM. 19% acceptance rate.

Geoffrey Belter, E. R. Jessup, Ian Karlin, and Jeremy G. Siek. Automating the generation of composed linear algebra kernels. In *International Conference for High Performance Computing, Networking, Storage and Analysis*, 2009. 20% acceptance rate.

Angela Yun Zhu, Walid Taha, Robert Cartwright, Matthieu Martel, and Jeremy G. Siek. In pursuit of real answers. In *Proceedings of the 2009 International Conference on Embedded Software and Systems*, pages 115–122, Washington, DC, USA, 2009. IEEE Computer Society. 29% acceptance rate.

Boyana Norris, Albert Hartono, Elizabeth Jessup, and Jeremy G. Siek. Generating empirically optimized composed matrix kernels from matlab prototypes. In *International Conference on Computational Science*, 2009. 30% acceptance rate.

Jeremy G. Siek, Ronald Garcia, and Walid Taha. Exploring the design space of higher-order casts. In *European Symposium on Programming*, March 2009. 27% acceptance rate.

Jeremy G. Siek and Manish Vachharajani. Gradual typing and unification-based inference. In *Dynamic Languages Symposium*. AITO, 2008. 43% acceptance rate.

Jeremy G. Siek and Walid Taha. Gradual typing for objects. In *ECOOP 2007*, volume 4609 of *LCNS*, pages 2–27. Springer Verlag, August 2007. 16% acceptance rate.

Seth Fogarty, Emir Pasalic, Jeremy G. Siek, and Walid Taha. Concoqtion: Indexed types now! In *ACM SIGPLAN 2007 Workshop on Partial Evaluation and Program Manipulation (PEPM '07)*, 2007. 55% acceptance rate.

Douglas Gregor, Jaakko Järvi, Jeremy G. Siek, Gabriel Dos Reis, Bjarne Stroustrup, and Andrew Lumsdaine. Concepts: Linguistic support for generic programming in C++. In *Proceedings of the ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications (OOPSLA'06)*, October 2006. 17% acceptance rate.

Jeremy G. Siek and Walid Taha. A semantic analysis of C++ templates. In *ECOOP 2006: European Conference on Object-Oriented Programming*, Nantes, France, July 2006. 13% acceptance rate.

Jaakko Järvi, Douglas Gregor, Jeremiah Willcock, Andrew Lumsdaine, and Jeremy G. Siek. Algorithm specialization in generic programming - challenges of constrained generics in C++. In *PLDI '06: Proceedings of the ACM SIGPLAN 2006 Conference on Programming Language Design and Implementation*, New York, NY, USA, June 2006. ACM Press. 21% acceptance rate.

Jeremy G. Siek and Andrew Lumsdaine. Essential language support for generic programming. In *PLDI '05: Proceedings of the ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation*, pages 73–84. ACM Press, June 2005. 21% acceptance rate.

Jeremy G. Siek and Andrew Lumsdaine. Language requirements for large-scale generic libraries. In *GPCE '05: Proceedings of the 4th International Conference on Generative Programming and Component Engineering*, September 2005. 29% acceptance rate.

Lie-Quan Lee, Jeremy G. Siek, and Andrew Lumsdaine. The generic graph component library. In *OOPSLA '99: Proceedings of the 14th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 399–414, New York, NY, USA, 1999. ACM Press. 18% acceptance rate.

Lie-Quan Lee, Jeremy G. Siek, and Andrew Lumsdaine. Generic graph algorithms for sparse matrix ordering. In *ISCOPE'99*, LNCS. Springer-Verlag, 1999. 34% acceptance rate.

Jeremy G. Siek and Andrew Lumsdaine. The matrix template library: A generic programming approach to high performance numerical linear algebra. In *ISCOPE'98*, LNCS. Springer-Verlag, 1998. 26% acceptance rate.

Refereed Workshop Papers

Christopher Schwaab and Jeremy G. Siek. Modular type-safety proofs in Agda. In *ACM SIGPLAN Workshop on Programming Languages meets Program Verification*, January 2013.

Jeremy G. Siek and Ronald Garcia. Interpretations of the gradually-typed lambda calculus. In *ACM SIGPLAN Workshop on Scheme and Functional Programming*, September 2012. 13 pages.

Geoffrey Belter, Jeremy G. Siek, Ian Karlin, and E. R. Jessup. Automatic generation of tiled and parallel linear algebra routines. In *Fifth International Workshop on Automatic Performance Tuning (iWAPT)*, June 2010.

Elizabeth R. Jessup, Ian Karlin, Erik Silkensen, Geoffrey Belter, and Jeremy Siek. Understanding memory effects in the automated generation of optimized matrix algebra kernels. In *Automated Program Generation Workshop at ICCS*, May 2010.

Justin E. Gottschlich, Jeremy G. Siek, Manish Vachharajani, Dwight Y. Winkler, and Daniel A. Connors. An efficient lock-aware transactional memory implementation. In *Proceedings of the 4th workshop on the Implementation, Compilation, Optimization of Object-Oriented Languages and Programming Systems*, ICOOOLPS '09, pages 10–17, New York, NY, USA, 2009. ACM.

Jeremy G. Siek and Philip Wadler. Threesomes, with and without blame. In *Proceedings for the 1st workshop on Script to Program Evolution*, STOP '09, pages 34–46, New York, NY, USA, 2009. ACM.

Justin E. Gottschlich, Jeremy G. Siek, and Daniel A. Connors. C++ move semantics for exception safety and optimization in software transactional memory libraries. In *Proceedings of the Third International Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems (ICOOOLPS). In conjunction with ECOOP.* jul 2008.

Jeremy G. Siek, Ian Karlin, and E. R. Jessup. Build to order linear algebra kernels. In *Workshop on Performance Optimization for High-Level Languages and Libraries (POHLL 2008)*, pages 1–8, April 2008.

Jeremy G. Siek and Walid Taha. Gradual typing for functional languages. In *Scheme and Functional Programming Workshop*, pages 81–92, September 2006.

Jaakko Järvi, Andrew Lumsdaine, Jeremy Siek, and Jeremiah Willcock. An Analysis of Constrained Polymorphism for Generic Programming. In Kei Davis and Jörg Striegnitz, editors, *Multiparadigm Programming 2003: Proceedings of the MPOOL Workshop at OOPSLA'03*, John von Neumann Institute of Computing series, pages 87–107, Anaheim, CA, October 2003.

Doug Gregor, Brian Osman, David R. Musser, Jeremy G. Siek, Lie-Quan Lee, and Andrew Lumsdaine. Concept-based component libraries and optimizing compilers. In *NSF Next Generation Systems Program Workshop at International Parallel and Distributed Processing Symposium, IPDPS 2002*, pages 174 –181, 2002.

David Abrahams and Jeremy G. Siek. Policy adaptors and the Boost Iterator Adaptor Library. In *Second Workshop on C++ Template Programming*, October 2001.

Jeremiah Willcock, Jeremy Siek, and Andrew Lumsdaine. Caramel: A concept representation system for generic programming. In *Second Workshop on C++ Template Programming*, October 2001.

Jeremy G. Siek and Andrew Lumsdaine. Concept checking: Binding parametric polymorphism in C++. In *Proceedings of the First Workshop on C++ Template Programming*, Erfurt, Germany, 2000.

Jeremy G. Siek and Andrew Lumsdaine. Mayfly: A pattern for lightweight generic interfaces. In *Pattern Languages of Programs*, July 1999.

Jeremy G. Siek, Andrew Lumsdaine, and Lie-Quan Lee. Generic programming for high performance numerical linear algebra. In *Proceedings of the SIAM Workshop on Object Oriented Methods for Inter-operable Scientific and Engineering Computing (OO'98)*. SIAM Press, 1998.

Jeremy G. Siek and Andrew Lumsdaine. The Matrix Template Library: A unifying framework for numerical linear algebra. In *Parallel Object Oriented Scientific Computing*. ECOOP, 1998.

Software

Build to Order BLAS. Available at `http://ecee.colorado.edu/wpmu/btoblas/`.

Peer-reviewed Boost libraries: Graph, Concept Check, Dynamic Bitset, Iterator Adaptor, Operator, and Property Map. The Boost Library Collection is available at `www.boost.org`.

The Matrix Template Library and the Iterative Template Library. The Matrix Template Library is available at `www.osl.iu.edu/research/mtl` and the Iterative Template Library is available at `www.osl.iu.edu/research/itl`.

Funding

Unifying Modular Abstractions for Chapel, Part 2. (NSF CAREER Award Supplement) Principal Investigator: Jeremy G. Siek. Funding Agency: DOD, Amount: $89,906, Duration: 8/27/2013–8/26/2014.

Modular Reflection. Principal Investigator: Bor-Yuh Chang, Co-PI: Jeremy G. Siek. Funding Agency: NSF. Amount: $ 125,000 (Siek's part), Duration: 10/1/2012–9/30/2015.

Unifying Modular Abstractions for Chapel. (NSF CAREER Award Supplement) Principal Investigator: Jeremy G. Siek. Funding Agency: DOD, Amount: $88,816, Duration: 8/27/2012–8/26/2013.

REU for CAREER: Bridging the Gap Between Prototyping and Production. Principal Investigator: Jeremy G. Siek. Funding Agency: NSF. Amount: $21,000, Duration: 6/1/2012–5/31/2013.

Modern Object-Oriented Chapel. (NSF CAREER Award Supplement) Principal Investigator: Jeremy G. Siek. Funding Agency: DOD, Amount: $39,905, Duration: 7/1/2011–6/30/2012.

Remodel ECEE 1B61A Graduate Student Office Space. Principal Investigator: Jeremy G. Siek. Funding Agency: CU CEAS and the CU ECEE Dept, Amount: $55,120, June 2011.

Interfaces and Modular Generics for Chapel. Principal Investigator: Jeremy G. Siek. Funding Agency: DOD, Amount: $103,735, Duration: 9/1/2010–8/31/2012.

EAGER: Exploratory Research on Gradual Programming. Principal Investigator: Jeremy G. Siek. Funding Agency: NSF. Amount: $81,748,
Duration: 8/1/2009–7/31/2010.

CAREER: Bridging the Gap Between Prototyping and Production. CCF 0846121. Principal Investigator: Jeremy G. Siek. Funding Agency: NSF. Amount: $481,910, Duration: 3/1/2009–2/28/2014.

REU for CAREER: Bridging the Gap Between Prototyping and Production. Principal Investigator: Jeremy G. Siek. Funding Agency: NSF. Amount: $10,500, Duration: 3/1/2009–2/28/2010.

Matching funds from the University of Colorado for the CAREER award. Amount: $60,000.

Test and Evaluation of Architecture-Aware Compiler Environments. Principal Investigator: Jeremy G. Siek. Funding Agency: DARPA. Amount: $1,146,195, Duration: 2/1/2009–7/1/2013.

Modular Metaprogramming. Principal Investigators: Jeremy G. Siek (Colorado) and Andrew Lumsdaine (Indiana). Funding Agency: NSF. Amount: $340,000 (Colorado's portion), Duration: 7/1/2007–7/1/2010.

Presentations

Invited Presentations

Reliable Generation of High-Performance Matrix Algebra. University of Utah. May 2013.

The gradual typing approach to mixing static and dynamic typing. Keynote at the Trends in Functional Programming Conference. May 2013.

Foundations for Gradually Typed Languages. Indiana University, February 2013.

Interpretations of the Gradually-Typed Lambda Calculus. Distilled Tutorial at the ACM SIGPLAN Workshop on Scheme and Functional Programming. September 2012.

Reliable Generation of High-Performance Matrix Algebra. NII Shonan Meeting on Bridging the Theory of Staged Programming and the Practice of High-Performance Computing. May 2012.

Gradual Typing Roundup. Dagstuhl Seminar on Foundations for Scripting Langauges. January 2012.

Composable DSLs. IFIP WG 2.11, September 2011.

Build to Order Linear Algebra Kernels. University of Glasgow, June 2010.

General Purpose Languages Should be Metalanguages. University of Edinburgh, Laboratory for Foundations in Computer Science, June 2010.

The C++0x Concept Effort. University of Washington. May, 2010.

Concepts in C++. Spring School on Generic and Indexed Programming. Oxford University, UK. March, 2010.

General Purpose Languages Should be Metalanguages. Keynote at the ACM SIGPLAN 2010 Workshop on Partial Evaluation and Program Manipulation. Madrid, Spain. January, 2010.

Build to Order Linear Algebra Kernels. NSF-NAIS Workshop, Intelligent Software: the interface between algorithms and machines. Edinburgh, UK. October, 2009.

Build To Order BLAS. CScADS Workshop on Libraries and Autotuning for Petascale Applications. Tahoe, CA. August, 2009.

Space-Efficient Blame Tracking for Gradual Typing. The University of Edinburgh, March 2009.

Type Safe Reflective Metaprogramming. Microsoft Research Redmond, RiSE team, December 2008.

Panel: The Future of Programming Languages. Microsoft's Professional Developers Conference, October 2008.

Gradual Typing with Inference. Foundations of Object-Oriented Languages (FOOL'08), January, 2008. San Francisco.

Knights' Tours, Religious Wars, and Built to Order BLAS. University of Colorado at Boulder, April 2007.

Build to Order Linear Algebra Kernels. Tech-X Corporation, March 2007.

Language Support for Generic Programming, C++200X and Beyond. Rensselaer Polytechnic Institute, March 2007.

Languages and Libraries for High-Performance Computing. U. of Wyoming, March 2007.

Gradual Typing. Oxford University, February 2007.

Languages and Libraries for High-Performance Computing. Oxford Univ., Nov. 2006.

Languages and Libraries for High-Performance Computing. University of Colorado at Boulder, May 2006.

Languages and Libraries for High-Performance Computing. University of California, Merced, May 2006.

Languages and Libraries for High-Performance Computing. Virginia Polytechnic Institute and State University, February 2006.

Languages and Libraries for High-Performance Computing. MathWorks, February 2006.

The Design and Implementation of the Boost Graph Library. The Association of C and C++ Users (ACCU) Spring Conference 2003.

A Language for Generic Programming. April, 2005. Rice University.

Modular Generics. The Adobe Systems Technical Summit, April 2004.

Generic Programming and Numerics. Center for Theoretical Studies, ETH Zürich. 2002.

Generic Software Components for Scientific Computing. The Institute for Scientific Computing Research at Lawrence Livermore National Lab. November 1999.

Conference Presentations

Effects for Funargs. ACM SIGPLAN Workshop on Higher-Order Programming with Effects, September 2012.

Monads for Relations. Scottish Programming Languages Seminar, June 2010.

Threesomes, With and Without Blame. Symposium on Principles of Programming Languages, January, 2010.

Blame Tracking for Gradual Types. JVM Language Summit, September, 2009. Sun Microsystems, Santa Clara, CA.

Threesomes, With and Without Blame. 1st International Workshop on Script to Program Evolution, July, 2009. Genoa, Italy.

Exploring the Design Space of Higher-Order Casts. European Symposium on Programming, March 2009. York, United Kingdom.

Gradual Typing for Python. JVM Language Summit, September, 2008. Sun Microsystems, Santa Clara, CA.

Build to Order Linear Algebra Kernels. Front Range Architecture Compilers Tools and Languages Workshop (FRACTAL), April, 2008.

Build to Order Linear Algebra Kernels. Joint Workshop on High-Level Parallel Programming Models and Supportive Environments and Performance Optimization for High-Level Languages and Libraries, April, 2008.

Gradual Typing with Inference. PC Meeting for the 2008 European Conference on Object-Oriented Programming. February, 2008.

Gradual Typing for Objects. European Conference on Object-Oriented Programming, July, 2007, Berlin.

Effectively writing about your research. ECOOP 2007 Doctoral Symposium and Ph.D. Students Workshop, July, 2007, Berlin.

Generic Programming and the Boost Graph Library. The Boost Libraries Conference. May 2007, Aspen.

Gradual Typing for Functional Languages. Scheme and Functional Programming Workshop, September 2006.

Language Requirements for Large-Scale Generic Libraries. The conference on Generative Programming and Component Engineering. September 2005.

Essential Language Support for Generic Programming. The ACM SIGPLAN 2005 conference on Programming Language Design and Implementation. June, 2005.

Modular Generics. The OOPSLA 2004 doctoral symposium.

Policy Adaptors and the Boost Iterator Adaptor Library. The Second Workshop on C++ Template Programming, October 2001.

Concept checking: Binding parametric polymorphism in C++. The First Workshop on C++ Template Programming, Erfurt, Germany, 2000.

Generic Programming for High Performance Numerical Linear Algebra. SciTools in Oslo, Norway. 1998. Award for best presentation.

The Matrix Template Library: A Generic Programming Approach to High Performance Numerical Linear Algebra. The *International Symposium on Computing in Object-Oriented Parallel Environments (ISCOPE)*. December 1998.

The Matrix Template Library: A Unifying Framework for Numerical Linear

Algebra. The Parallel and High-Performance Object-Oriented Computing workshop. July 1998.

Generic Programming for High Performance Numerical Linear Algebra. The SIAM Workshop on OO Methods for Interoperable Scientific and Engineering Computing, 1998.

Teaching

For Faculty Course Questionnaires (FCQ), maximum rating is 6.

Undergraduate Courses

1. ECEN 2703: Discrete Mathematics for Computer Engineers, Fall 2011. FCQ course: 3.6, instructor: 4.2, availability: 4.9, workload: 7-9 hours/week, respect: 5.9.
2. ECEN 2703: Discrete Mathematics for Computer Engineers, Spring 2011. FCQ course: 3.5, instructor: 4.4, availability: 4.9, workload: 7-9 hours/week, respect: 5.9.
3. ECEN 3707: Discrete Mathematics for Computer Engineers, Spring 2010. FCQ course: 3.3, instructor: 3.4, availability: 4.2, workload: 7-9 hours/week, respect: 5.5.
4. CSCI 2830: Computer Science as a field of work and study. Fall 2006. FCQ course: 5.2, instructor: 5.7, availability: 5.8, workload: 0-3 hours/week.

Cross-listed Graduate and Undergraduate Courses

1. ECEN 4553/5523 and CSCI 4555/5525: Compiler Construction, Fall 2012. FCQ course: 4.9, instructor: 5.2, availability: 5.0, workload: 13-15 hours/week, respect: 5.8.
2. ECEN 4553/5523 and CSCI 4555/5525: Compiler Construction, Fall 2010. FCQ course: 5.8, instructor: 5.7, availability: 5.6, workload: 13-15 hours/week, respect: 5.9.
3. ECEN 4553/5523 and CSCI 4555/5525: Compiler Construction, Fall 2009. FCQ course: 4.3, instructor: 4.6, availability: 4.8, workload: 13-15 hours/week, respect: 5.9.
4. ECEN 4553/5013: Introduction to Compiler Construction, Fall 2008. FCQ course: 4.0, instructor: 4.8, availability: 5.2, workload: 13-15 hours/week, respect: 5.8.
5. ECEN 4553/5013: Introduction to Compiler Construction, Fall 2007. FCQ course: 5.2, instructor: 5.2, availability: 5.7, workload: 16+ hours/week, treatment: 5.9.
6. CSCI 4448/6448: Object-Oriented Analysis and Design. Spring 2007. FCQ course: 3.1, instructor: 3.4, availability: 4.3, workload: 7-9 hours/week.

Graduate Courses

1. ECEN 5533: Fundamentals of Programming Languages, Fall 2011. FCQ course: 4.6, instructor: 5.1, availability: 5.4, workload: 10-12 hours/week, respect: 6.0.
2. ECEN 5013: Theorem Proving in Isabelle, Spring 2011. FCQ course: 5.0, instructor: 5.5, availability: 5.0, workload: 10-12 hours/week, respect: 5.8.
3. ECEN 5013: Types and Programming Languages, Spring 2010. FCQ course: 5.2, instructor: 5.4, availability: 5.0, workload: 4-6 hours/week, respect: 5.8.
4. ECEN 5043: Software Engineering Reusable Components, Spring 2009. FCQ course: 4.4, instructor: 5.1, availability: 4.9, workload: 4-6 hours/week, respect: 5.8.
5. ECEN 5023/CSCI 7135-001: Types and Programming Languages, Spring 2008. FCQ course: 5.0, instructor: 5.3, availability: 5.0, workload: 7-9 hours/week, respect: 5.5.
6. CSCI 7000: Practical Theorem Proving with Isabelle/Isar. Spring 2007.

Short Courses and Tutorials

1. C++, Short and Sweet. Online video lecture available at Udemy.
2. Generic Programming and the Boost Graph Library. May 14, 2007. Tutorial at the Boost Libraries Conference 2007.
3. Generic Programming and the Boost Libraries. Short course taught at Engineering Dynamics, Inc. New Orleans. March 21-28, 2006.
4. The Practice of Generic Programming. With Jaakko Järvi. Invited tutorial at the Adobe Systems Technical Summit, April 2004.

Advising

- **Current Ph.D. Students**
  - Thomas Nelson
  - Weiyu Miao
  - John Michalakes
  - Michael Vitousek
  - Chris Wailes
- **Current B.S. Students**
  - Xing Jie Zhong
- **Co-advising**
  - Devin Coughlin, Ph.D. student.
- **Graduated Students, Advised**
  - Shashank Bharadwaj, M.S. Fall 2012, now at Amazon.
  - Neelam Agrawal, M.S. Spring 2012, now at National Instruments.
  - Geoffrey Belter, Ph.D. Fall 2012, now at Apple.

  - Justin E. Gottschlich, Ph.D, Fall 2010, now at Intel Labs.
  - Jonathan Turner, M.S. Fall 2012, now at Microsoft.
  - Erik Silkensen, M.S. Spring 2012, pursuing a Ph.D. at Northeastern.
  - Moss Prescott, M.S. Fall 2010.
  - Sri Teja Basava, M.S. Spring 2011, now at National Instruments.
  - Christopher Schwaab, B.S. Spring 2010.
- **Graduated Students, Co-advised**
  - Ian Karlin, Ph.D. student, Spring 2011.
  - Scott Busch, M.S. Thesis, Fall 2009.
  - David Broman, Ph.D. at Linköping University, Spring 2010.
  - Tipp Moseley, Ph.D., Spring 2009. Now at Google.
  - Scott Williams, B.S., Independent Study Senior Project, Fall 2006.
  - Jeff Fifield, Ph.D. student, Fall 2011.

Mentoring

- Google Summer of Code, Summer 2006, 2007, 2008.

Service

Department

- Departmental Executive Committee AY 2009–2012
- Computer Engineering Search Committee AY 2011-2012.
- Head of the Digital Core Curriculum Committee AY 2012–2013.

Research Community

- PC Member. POPL 2014: The ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages.
- PC Member. The International Conference on Generative Programming: Concepts & Experiences (GPCE 2013).
- PC Member. The ACM SIGPLAN 2013 Workshop on Partial Evaluation and Program Manipulation (PEPM).
- PC Member. The ACM 2013 Conference on Programming Language Design and Implementation (PLDI).
- Review Committee Member. 2013 ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL).
- PC Member. NIER track of the 2012 International Conference on Software Engineering (ICSE).
- PC Member. Generative Programming and Component Engineering (GPCE) 2011.

- PC Member. DSL 2011: IFIP Working Conference on Domain-Specific Languages.
- PC Member. The International Workshop on Languages and Compilers for Parallel Computing (LCPC 2011).
- PC Member. Onward! 2011.
- Review Committee Member. ACM Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA 2011).
- Program Chair. 2011 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation (PEPM 2011).
- Program Chair. 2010 International Workshop on Foundations of Object-Oriented Languages (FOOL 2010).
- PC Member. The Fifth international Workshop on Automatic Performance Tuning (iWAPT 2010).
- PC Member. The ACM SIGPLAN Workshop on Types in Language Design and Implementation (TLDI 2010).
- PC Member. 1st International Workshop on Script to Program Evolution (STOP 2009).
- General Chair. Generative Programming and Component Engineering (GPCE) 2009.
- PC Member. The ACM 2009 Conference on Programming Language Design and Implementation (PLDI).
- PC Member. IFIP Working Conference on Domain Specific Languages (DSL WC), July 2009.
- PC Member. ACM Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA 2008)
- Program Chair. Generative Programming and Component Engineering (GPCE) 2008.
- PC Member. ACM SIGPLAN 2008 Workshop on Partial Evaluation and Program Manipulation (PEPM 2008).
- PC Member. First Workshop on Generative Technologies (WGT 2008).
- PC Member. European Conference on Object-Oriented Programming (ECOOP 2008).
- Panel member for the ECOOP 2007 Doctoral Symposium and Ph.D. Students Workshop.
- PC Member. Generative Programming and Component Engineering (GPCE) 2007.
- PC Member. The Boost Libraries Conference 2007.
- Co-Chair. ACM SIGPLAN 2007 Symposium on Library-Centric Software Design. Co-located with OOPSLA. I was responsible for obtaining ACM sponsorship.

- PC Member. Workshop on Generic Programming at ICFP 2006.
- Organizer. Library-Centric Software Design Workshop at OOPSLA 2005 and 2006.
- PC Member. Parallel Object-Oriented Scientific Computing Workshop at ECOOP 2005 and 2006.
- PC Member. The Second MetaOCaml Workshop at GPCE 2005.
- Reviewer. ESOP 2007, PARA 2006, POPL 2006, 2009, and 2011, PEPM 2009, ACM TOPLAS, ACM TOMS, ICFP 2003, Software Practice and Experience, Wiley Encyclopedia of Computer Science and Education (2008), Science of Computer Programming, Journal of Object Technology, Journal of Functional Programming..

Committees and Editing

- Associate Editor, ACM TOMS 2011–present
  Transactions on Mathematical Software
- GPCE Steering Committee 2008–present
  International Conference on Generative Programming and Component Engineering.
- FOOL Steering Committee (Chair) 2011–present
  International Workshop on Foundations of Object-Oriented Languages.
- PEPM Steering Committee 2011–present
  Workshop on Partial Evaluation and Program Manipulation.
- Guest Editor. Special issue of the journal Science of Computer Programming on Library-Centric Software Design.
- C++ ANSI/ISO Standards Committee 2001–2008
  Worked on the proposal for adding support for generic programming to C++ through the addition of special kinds of interfaces called "concepts". Served in the Library Working Group evaluating extensions to the C++ Standard Library. Worked on the iterator concept and iterator adaptor proposals for standard library extension.

Other

- Contributor to Boost C++ Open Source Group 1999–2005
  Contributed and maintained many software libraries including the Boost Graph Library. Served as review manager for the Boost Unit Test Library, the Boost Preprocessor Library, and the Boost MPI Library.