

Jeremy Graham Siek

Dept. of Electrical, Computer, and Energy Engineering
University of Colorado at Boulder
425 UCB
Boulder, CO 80309-0425

jeremy.siek@colorado.edu
Birth date: Jan. 8, 1975
Nationality: USA

Interests Research in language design and implementation to improve support for software libraries and domain specific languages, including research in type systems, program logics, facilities for program generation, and high-level program optimization. The design of software libraries using generic programming for domains such as high-performance linear algebra and applied graph theory.

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

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

University of Notre Dame B.S. Mathematics 1997.

Experience **University of Colorado at Boulder** 2007–present
Assistant Professor Boulder, CO
Researching programming language support for generic programming, metaprogramming, and the integration of static and dynamic type systems through gradual typing. Developing an optimizing compiler for MATLAB. Teaching courses on compilers, type systems, and generic programming.

LogicBlox 2006–2007
Research Scientist Atlanta, GA
Researched programming languages and run-time systems for deductive databases, in particular, extending Datalog with an advanced type system and support for program generation.

University of Colorado at Boulder 2006–2007
Visiting Assistant Professor Boulder, CO
Extended gradual typing to object-oriented languages and wrote a prototype of an optimizing compiler for MATLAB. Taught courses on object-oriented design and practical theorem proving with the Isabelle/Isar system.

Rice University 2005–2006
Post-doctoral Research Associate Houston, TX
Worked on language support for program generation with Walid Taha. I developed a formal semantics for C++ templates and integrated the MetaOCaml programming language with the Coq theorem prover. Also developing a type system that provides a gradual transition between dynamic and static typing.

Indiana University 2000–2005
Research Assistant Bloomington, IN
Worked in the Open System Laboratory directed by Andrew Lumsdaine on programming language support for generic programming and developed generic software libraries such as the Boost Graph Library.

AT&T Labs—Research*Summer Manager*

Summer 2001

Florham Park, NJ

Worked on the Extended Type Information (XTI) library, a system for compile-time reflection of type information for C++, with Bjarne Stroustrup. Applied XTI to the construction of a remote-procedure invocation system.

Silicon Graphics Inc.*Intern*

1999–2000

Mountain View, CA

Worked on the port of the SGI iostream library to Linux. Developed the concept checking library in collaboration with Alexander Stepanov and Matthew Austern. Contributed to bootstrapping the SGI C++ compiler in the IA64 Itanium processor.

Awards**NSF Faculty Early Career Development (CAREER) Award***Bridging the Gap Between Prototyping and Production*, March 2009.

Distinguished Visiting Fellowship, Scottish Informatics & Computer Science Alliance (SICSA), May-June, 2010.

PUBLICATIONS**Books**

The Boost Graph Library: User Guide and Reference. C++ In-Depth Series, Addison-Wesley. Jeremy G. Siek, Lee-Quan Lee, and Andrew Lumsdaine. December 20, 2001.

Book Chapters

A Modern Framework for Portable High Performance Numerical Linear Algebra. Jeremy G. Siek and Andrew Lumsdaine. In *Modern Software tools for Scientific Computer*. Birkhauser 1999.

Journal Articles

1. **SCP**: A Language for Generic Programming in the Large. Jeremy G. Siek and Andrew Lumsdaine. *Science of Computer Programming*, Article in Press, September 2008.
2. **HOSC**: Improving the Lazy Krivine Machine. Daniel P. Friedman, Abdulaziz Ghuloum, Jeremy G. Siek, and Lynn Winebarger. *Higher-Order and Symbolic Computation*, Volume 20, Number 3, September, 2007.
3. **JFP**: An extended comparative study of language support for generic programming. Ronald Garcia, Jaako Järvi, Andrew Lumsdaine, Jeremy Siek, and Jeremiah Willcock. *Journal of Functional Programming*, Volume 17, Issue 2, March 2007, pp 145-205.

Competitive Conference Papers

Acceptance rates are given in parenthesis after the conference acronym.

1. **CGO'10 (39%)**: An Efficient Software Transactional Memory Using Commit-Time Invalidation. Justin E. Gottschlich, Manish Vachharajani, and Jeremy G. Siek. In the *International Symposium on Code Generation and Optimization*, April 2010.
2. **POPL'10 (19%)**: Threesomes, With and Without Blame. Jeremy G. Siek and Philip Wadler. In the *Symposium on Principles of Programming Languages*, January 2010.
3. **SC'09 (20%)**: Automating the Generation of Composed Linear Algebra Kernels. Geoffrey Belter, E. R. Jessup, Ian Karlin, and Jeremy G. Siek. In the *International Conference on High Performance Computing, Networking, Storage and Analysis*, November 2009.
4. **ICISS'09 (23%)**: In Pursuit of Real Answers. Angela Yun Zhu, Walid Taha, Robert Cartwright, Matthieu Martel, and Jeremy G. Siek. In the *International Conference on Embedded Software and Systems*, May 2009.
5. **ESOP'09 (27%)**: Exploring the Design Space of Higher-Order Casts. Jeremy G. Siek, Ronald Garcia, and Walid Taha. In the *European Symposium on Programming*, March 2009.

6. **ECOOP'07 (16%)**: Gradual Typing for Objects. Jeremy G. Siek and Walid Taha. In the *21st European Conference on Object-Oriented Programming*, July 2007.
7. **OOPSLA'06 (17%)**: Concepts: Linguistic Support for Generic Programming in C++. Douglas Gregor, Jaakko Järvi, Jeremy G. Siek, Gabriel Dos Reis, Bjarne Stroustrup, and Andrew Lumsdaine. In the *ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, October 2006.
8. **ECOOP'06 (13%)**: A Semantic Analysis of C++ Templates. Jeremy G. Siek and Walid Taha. In the *European Conference on Object-Oriented Programming*, July 2006.
9. **PLDI'06 (21%)**: Algorithm specialization in generic programming - Challenges of constrained generics in C++. Jaakko Järvi, Douglas Gregor, Jeremiah Willcock, Andrew Lumsdaine, and Jeremy G. Siek. In the *ACM SIGPLAN 2006 Conference on Programming Language Design and Implementation*, June 2006.
10. **PLDI'05 (21%)**: Essential Language Support for Generic Programming. Jeremy Siek and Andrew Lumsdaine. In the *ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation*, June 2005.
11. **OOPSLA'03 (18%)**: A Comparative Study of Language Support for Generic Programming. Ronald Garcia, Jaakko Järvi, Andrew Lumsdaine, Jeremy G. Siek, and Jeremiah Willcock. In the *2003 ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, October 2003.
12. **OOPSLA'99 (20%)**: The Generic Graph Component Library. Lie-Quan Lee, Jeremy G. Siek, and Andrew Lumsdaine. In the *ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, pages 399–414, October 1999.

**Refereed
Conference
and
Workshop
Papers**

1. An Efficient Lock-Aware Transactional Memory Implementation. Justin E. Gottschlich, Jeremy G. Siek, Manish Vachharajani, Dwight Y. Winkler, and Daniel A. Connors. In the *4th International Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems (ICOOOLPS'2009)*.
2. Threesomes, With and Without Blame. Jeremy G. Siek and Philip Wadler. In the *1st International Workshop on Script to Program Evolution (STOP 2009)*, July 2009.
3. Generating Empirically Optimized Composed Matrix Kernels from MATLAB Prototypes. Boyana Norris, Albert Hartono, Elizabeth Jessup, Jeremy G. Siek. In the *International Conference on Computational Science 2009*, May 2009.
4. Gradual Typing with Unification-based Inference. Jeremy G. Siek and Manish Vachharajani. In the *Dynamic Languages Symposium*, July 2008.
5. C++ Move Semantics for Exception Safety and Optimization in Software Transactional Memory Libraries. Justin Gottschlich, Jeremy G. Siek, and Daniel A. Connors. In the *3rd International Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems (ICOOOLPS'2008)*.
6. Build to Order Linear Algebra Kernels. Jeremy Siek, Ian Karlin, and E. R. Jessup. In the *Workshop on Performance Optimization of High-level Languages and Libraries (POHLL 2008)*.
7. Concoction: Indexed Types Now! Seth Fogarty, Emir Pasalic, Jeremy Siek, and Walid Taha. In the *2007 ACM SIGPLAN workshop on Partial Evaluation and Program Manipulation (PEPM 2007)*.
8. Gradual typing for functional languages. Jeremy G. Siek and Walid Taha. *Scheme and Functional Programming Workshop 2006* at ICFP, September 2006.
9. Language Requirements for Large-Scale Generic Libraries. Jeremy Siek and Andrew Lumsdaine. In *Generative Programming and Component Engineering*. September, 2005. Volume 3676 of *Lecture Notes in Computer Science*, Springer-Verlag.

10. An Analysis of Constrained Polymorphism for Generic Programming. Jaakko Järvi, Andrew Lumsdaine, Jeremy Siek, and Jeremiah Willcock. In the *Multiparadigm Programming in Object-Oriented Languages Workshop (MPOOL)* at OOPSLA, October 2003.
11. Concept-Based Component Libraries and Optimizing Compilers. Sibylle Schupp, D. P. Gregor, B. Osman, David R. Musser, Jeremy G. Siek, Lie-Quan Lee, and Andrew Lumsdaine. In the NSF Next Generation Systems Program Workshop at IPDPS, 2002.
12. Policy Adaptors and the Boost Iterator Adaptor Library. David Abrahams and Jeremy Siek. In the *Second Workshop on C++ Template Programming*, October 2001.
13. Caramel: A Concept Representation System for Generic Programming. Jeremiah Willcock, Jeremy Siek, and Andrew Lumsdaine. In the *Second Workshop on C++ Template Programming*, Tampa, Florida, October 2001.
14. Concept checking: binding parametric polymorphism in C++. Jeremy G. Siek and Andrew Lumsdaine. In the *First Workshop on C++ Template Programming*. October 10, 2000.
15. Generic Graph Algorithms for Sparse Matrix Ordering. Lie-Quan Lee, Jeremy G. Siek, and Andrew Lumsdaine. In the *International Symposium on Computing in Object-Oriented Parallel Environments*, Volume 1732 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.
16. Mayfly: A Pattern for Lightweight Generic Interfaces. Jeremy G. Siek and Andrew Lumsdaine. In *Pattern Languages of Programs*, July 1999.
17. The Matrix Template Library: A Generic Programming Approach to High Performance Numerical Linear Algebra. Jeremy G. Siek and Andrew Lumsdaine. In the *International Symposium on Computing in Object-Oriented Parallel Environments*, Volume 1505 of *Lecture Notes in Computer Science*, pages 59–70. Springer-Verlag, 1998.
18. Generic Programming for High Performance Numerical Linear Algebra. Jeremy G. Siek, Andrew Lumsdaine, and Lie-Quan Lee. In *Proceedings of the SIAM Workshop on OO Methods for Interoperable Scientific and Engineering Computing (OO'98)*, 1998. SIAM Press.
19. The Matrix Template Library: A Unifying Framework for Numerical Linear Algebra. Jeremy G. Siek and Andrew Lumsdaine. In the *Workshop on Parallel Object Oriented Scientific Computing*, 1998. ECOOP.

Invited papers

Modular Generics. Jeremy G. Siek and Andrew Lumsdaine. In *Concepts: a Linguistic Foundation of Generic Programming*, April 2004. Adobe Systems.

C++ Standards Committee Reports

1. Scoped Concept Maps. ISO/IEC JTC 1, SC 22, Programming Language C++. N2098=06-0168. 2006.
2. Implementing Concepts. Douglas Gregor and Jeremy G. Siek. ISO/IEC JTC 1, SC 22, Programming Language C++. N1848=05-0108. 2005.
3. Explicit model definitions are necessary. Doug Gregor and Jeremy G. Siek. ISO/IEC JTC 1, SC 22, Programming Language C++. N1798=05-0058. 2005.
4. Concepts for C++0x, Revision 1. Douglas Gregor, Jeremy Siek, Ronald Garcia, Jeremiah Willcock, Jaakko Järvi, and Andrew Lumsdaine. ISO/IEC JTC 1, SC 22, Programming Language C++. N1849=05-0109. 2005.
5. Concepts for C++0x. Jeremy Siek, Douglas Gregor, Ronald Garcia, Jeremiah Willcock, Jaakko Järvi, and Andrew Lumsdaine. ISO/IEC JTC 1, SC 22, Programming Language C++. N1758=05-0018. 2005.
6. New Iterator Concepts. Jeremy Siek, David Abrahams, and Thomas Witt. ISO/IEC JTC 1, SC 22, Programming Language C++. N1477=03-0060. 2003.
7. Iterator Facade and Adaptor. David Abrahams, Jeremy Siek, and Thomas Witt. ISO/IEC JTC 1, SC 22, Programming Language C++. N1476=03-0059. 2003.

8. Decltype and Auto. Jaakko Järvi, Bjarne Stroustrup, Douglas Gregor, and Jeremy Siek. ISO/IEC JTC 1, SC 22, Programming Language C++. N1478=03-0061. 2003.

Journal and Magazine Articles

1. The Matrix Template Library: Generic Components for High-performance Scientific Computing. Jeremy G. Siek and Andrew Lumsdaine. *Computing in Science and Engineering*, 1(6):70–78, November 1999.
2. C++ Concept Checking. Jeremy G. Siek and Andrew Lumsdaine. *Dr. Dobb's Journal*, June 2001.
3. The Generic Graph Component Library. Jeremy G. Siek and Andrew Lumsdaine. *Dr. Dobb's Journal*, September 2000.
4. Software Engineering for Peak Performance. Jeremy G. Siek and Andrew Lumsdaine. *C++ Report*, pp 23–27, May 2000.

Technical Reports

1. Gradual Typing for Objects: Isabelle Formalization. Jeremy G. Siek Walid Taha. December 2006.
2. Concoqtion: Mixing Indexed Types and Hindley-Milner Type Inference. Emir Pasalic, Jeremy Siek, and Walid Taha. July 2006.
3. C++.T Formalization in Isar. Jeremy G. Siek and Walid Taha. Rice University, December 2005.
4. Essential Language Support for Generic Programming: Formalization Part 1. Jeremy G. Siek and Andrew Lumsdaine. Technical report 605, Indiana University, December 2004.
5. Combining Optimizations, Combining Theories. Todd L. Veldhuizen and Jeremy G. Siek. Technical report 582, Indiana University, May 2003.
6. Using ParentheC to Transform Scheme Programs to C. Ronald Garcia, Jeremy G. Siek, Ruj Akavipat, Samuel Chun, David W. Mack, Heather Roinestad, and Daniel P. Friedman. December 2003.
7. A Rational Approach to Portable High Performance: The Basic Linear Algebra Instruction Set (BLAIS) and the Fixed Algorithm Size Template (FAST) Library. Jeremy G. Siek and Andrew Lumsdaine. 1998. TR 1998-25. Dept. of Computer Science and Engineering. University of Notre Dame.

FUNDING

- EAGER: Exploratory Research on Gradual Programming. Funding Agency: NSF. Amount: \$81,748, Duration: 8/1/2009–7/31/2010.
- CAREER: Bridging the Gap Between Prototyping and Production. Principal Investigator: Jeremy G. Siek. Funding Agency: NSF. Amount: \$481,910, Duration: 3/1/2009–2/28/2014.
- Matching funds from the University of Colorado for the CAREER award. \$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.

SOFTWARE

- 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.

INVITED TALKS

1. Build to Order Linear Algebra Kernels. NSF-NAIS Workshop, Intelligent Software: the interface between algorithms and machines. Edinburgh, UK. October, 2009.
2. Build To Order BLAS. CScADS Workshop on Libraries and Autotuning for Petascale Applications. Tahoe, CA. August, 2009.
3. Space-Efficient Blame Tracking for Gradual Typing. The University of Edinburgh, March 2009.
4. Type Safe Reflective Metaprogramming. Microsoft Research Redmond, RiSE team, December 2008.
5. Panel: The Future of Programming Languages. Microsoft's Professional Developers Conference, October 2008.
6. Gradual Typing with Inference. Foundations of Object-Oriented Languages (FOOL'08), January, 2008. San Francisco.
7. Knights' Tours, Religious Wars, and Built to Order BLAS. University of Colorado at Boulder, April 2007.
8. Build to Order Linear Algebra Kernels. Tech-X Corporation, March 2007.
9. Language Support for Generic Programming, C++200X and Beyond. Rensselaer Polytechnic Institute, March 2007.
10. Languages and Libraries for High-Performance Computing. U. of Wyoming, March 2007.
11. Gradual Typing. Oxford University, February 2007.
12. Languages and Libraries for High-Performance Computing. Oxford Univ., Nov. 2006.
13. Languages and Libraries for High-Performance Computing. University of Colorado at Boulder, May 2006.
14. Languages and Libraries for High-Performance Computing. University of California, Merced, May 2006.
15. Languages and Libraries for High-Performance Computing. Virginia Polytechnic Institute and State University, February 2006.
16. Languages and Libraries for High-Performance Computing. MathWorks, February 2006.
17. The Design and Implementation of the Boost Graph Library. The Association of C and C++ Users (ACCU) Spring Conference 2003.
18. A Language for Generic Programming. April, 2005. Rice University.
19. Modular Generics. The Adobe Systems Technical Summit, April 2004.
20. Generic Programming and Numerics. Center for Theoretical Studies, ETH Zürich. 2002.
21. Generic Software Components for Scientific Computing. The Institute for Scientific Computing Research at Lawrence Livermore National Lab. November 1999.

TALKS

1. Blame Tracking for Gradual Types. JVM Language Summit, September, 2009. Sun Microsystems, Santa Clara, CA.
2. Threesomes, With and Without Blame. 1st International Workshop on Script to Program Evolution, July, 2009. Genoa, Italy.
3. Exploring the Design Space of Higher-Order Casts. European Symposium on Programming, March 2009. York, United Kingdom.

4. Gradual Typing for Python. JVM Language Summit, September, 2008. Sun Microsystems, Santa Clara, CA.
5. Build to Order Linear Algebra Kernels. Front Range Architecture Compilers Tools and Languages Workshop (FRACTAL), April, 2008.
6. 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.
7. Gradual Typing with Inference. PC Meeting for the 2008 European Conference on Object-Oriented Programming. February, 2008.
8. Gradual Typing for Objects. European Conference on Object-Oriented Programming, July, 2007, Berlin.
9. Effectively writing about your research. ECOOP 2007 Doctoral Symposium and Ph.D. Students Workshop, July, 2007, Berlin.
10. Generic Programming and the Boost Graph Library. The Boost Libraries Conference. May 2007, Aspen.
11. Gradual Typing for Functional Languages. Scheme and Functional Programming Workshop, September 2006.
12. Language Requirements for Large-Scale Generic Libraries. The conference on Generative Programming and Component Engineering. September 2005.
13. Essential Language Support for Generic Programming. The ACM SIGPLAN 2005 conference on Programming Language Design and Implementation. June, 2005.
14. Modular Generics. The OOPSLA 2004 doctoral symposium.
15. Policy Adaptors and the Boost Iterator Adaptor Library. The Second Workshop on C++ Template Programming, October 2001.
16. Concept checking: Binding parametric polymorphism in C++. The First Workshop on C++ Template Programming, Erfurt, Germany, 2000.
17. Generic Programming for High Performance Numerical Linear Algebra. SciTools in Oslo, Norway. 1998. Award for best presentation.
18. 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.
19. The Matrix Template Library: A Unifying Framework for Numerical Linear Algebra. The Parallel and High-Performance Object-Oriented Computing workshop. July 1998.
20. 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.

1. 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.
2. 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.
3. 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.
4. Generic Programming and the Boost Graph Library. May 14, 2007. Tutorial at the Boost Libraries Conference 2007.
5. 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.
6. CSCI 7000: Practical Theorem Proving with Isabelle/Isar. Spring 2007.
7. 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.

8. Generic Programming and the Boost Libraries. Short course taught at Engineering Dynamics, Inc. New Orleans. March 21-28, 2006.
9. The Practice of Generic Programming. With Jaakko Järvi. Invited tutorial at the Adobe Systems Technical Summit, April 2004.

SERVICE

Advising

- Joe Angell, M.S. student.
- Geoffrey Belter, Ph.D. student.
- Weiyu Miao, Ph.D. student.
- Justin E. Gottschlich, Ph.D. student.
- Jay Kominek, M.S. student.
- Christopher Schwaab, Undergraduate Independent Study, Spring 2009.
- Google Summer of Code, Summer 2006, 2007, 2008.

Co-advising

- Ian Karlin, Ph.D. student.
- Tipp Moseley, Ph.D. student.
- David Broman, Ph.D. student at Linköping University.
- Scott Williams, Independent Study Senior Project, Fall 2006.

Conference and Journal Organization and Review

- 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.
- Guest Editor. Special issue of the journal Science of Computer Programming on Library-Centric Software Design.
- 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, ACM TOPLAS, ACM TOMS, ICFP 2003, Software Practice and Experience, Wiley Encyclopedia of Computer Science and Education (2008), Science of Computer Programming.

C++ Standards Committee 2001–present
 Worked on a 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.

Boost C++ Open Source Group 1999–present
 Contributed and maintaining many software libraries. Served as review manager for the Boost Unit Test Library, the Boost Preprocessor Library, and the Boost MPI Library.

On the advisory board of the online journal *The C++ Source* 2005–present

GPCE Steering Committee 2008–present
 International Conference on Generative Programming and Component Engineering.