Curriculum Vitae
Jonathan Aldrich

Contact Information

Jonathan Aldrich
Institute for Software Research
School of Computer Science
Carnegie Mellon University
4665 Forbes Avenue
Pittsburgh, PA 15213-3891

email: jonathan.aldrich@cs.cmu.edu
web: http://www.cs.cmu.edu/~aldrich/
phone: +1-412-268-7278
fax: +1-412-268-2338
office: 422 TCS Hall

Executive assistant:

Linda Campbell
lv2c at cs dot cmu dot edu

Research: Engineering Languages

I work at the intersection of **programming languages** and **software engineering**. My research examines new ways to **express software and its properties** that improve our ability to **engineer software at scale**. Effective software engineering at scale is closely tied to design---how a system is broken into parts, and how those parts compose to achieve the desired functionality and properties of the system. Thus, my research develops new ways to **express design within source code,** where both tools and engineers can most effectively leverage it, thereby improving productivity and reducing errors. My work also focuses on improved **object models**---a foundational composition mechanism---as well as **type systems and logics** for specifying component boundaries and reasoning about the result of composition. I evaluate the systems I develop using a wide variety of techniques, including **mathematical proofs, case studies, code corpus studies, and evaluations with human subjects**. One might say that I work on languages for better software engineering, but that I also take an **engineering approach to language design**: thinking not just about what a language can express, but the cost-benefit tradeoffs of various language constructs and how those constructs work together to help engineers develop software more effectively.

Education

Ph.D., Computer Science and Engineering, University of Washington, August 2003.
Advisors: Craig Chambers and David Notkin
Thesis: Using Types to Enforce Architectural Design

M.S., Computer Science and Engineering, University of Washington, June 1999.

B.S., Engineering and Applied Science (Computer Science), California Institute of Technology, June 1997.

Employment

2017-present	Professor, Carnegie Mellon University
2009-2017	Associate Professor, Carnegie Mellon University
2003-2009	Assistant Professor, Carnegie Mellon University
1997-2003	Graduate Student and Research Assistant, University of Washington
Summer 1997	Research Assistant, California Institute of Technology
Summers 1993-1996	Summer Intern, Sequent Computer Systems, Inc.

Selected Honors

Member, IFIP Working Group 2.16 on Programming Language Design, 2014-present

2012 ICSE Most Influential Paper Award, given for the paper ArchJava: Connecting Software Architecture to Implementation, by Jonathan Aldrich, Craig Chambers, and David Notkin, from ICSE 2002.

2007 Dahl-Nygaard Junior Prize (press release), an international award given annually for a significant technical contribution to object-oriented programming.

2007 DARPA Computer Science Study Group

2006 National Science Foundation CAREER award, "Lightweight Modeling and Enforcement of Architectural Behavior"

2003 William Chan Memorial Dissertation Award, University of Washington, Dept. Computer Science and Engineering

1997-2000 National Defense Science and Engineering Graduate Fellowship

1997-2000 Achievement Rewards for College Scientists Fellowship

1997 National Science Foundation Fellowship Honorable Mention

1995-1997 Caltech Merit Scholarship

1993 National Merit Scholarship

1996 Winner, Caltech-Occidental Symphony Concerto Competition (violin)

Honor societies: Sigma Xi (scientific research), Tau Beta Pi (engineering)

Publications

Book Chapters

1. Checking Concurrent Typestate with Access Permissions in Plural: A Retrospective. Kevin Bierhoff, Nels E. Beckman and Jonathan Aldrich. In Peri L. Tarr and Alexander L. Wolf, editors, *Engineering of Software: The Continuing Contributions of Leon J. Osterweil*, pages 35-48. Springer, 2011.
2. Practical Exception Specifications. Donna Malayeri and Jonathan Aldrich. In Christophe Dony, Jørgen Lindskov Knudsen, Alexander B. Romanovsky, and Anand Tripathi, editors, *Advanced Topics in Exception Handling Techniques*, volume 4119 of Lecture Notes in Computer Science, pages 200-220. Springer, 2006.

Refereed Journal Publications

1. PLIERS: A Process that Integrates User-Centered Methods into Programming Language Design. Michael Coblenz, Gauri Kambhatla, Paulette Koronkevich, Jenna L. Wise, Celeste Barnaby, Joshua Sunshine, Jonathan Aldrich, and Brad A. Myers. ACM Trans. Computer-Human Interaction (TOCHI) 28, 4, Article 28, August 2021.
2. Obsidian: Typestate and Assets for Safer Blockchain Programming. Michael Coblenz, Reed Oei, Tyler Etzel, Paulette Koronkevich, Yannick Bloem, Brad A. Myers, Joshua S. Sunshine, and Jonathan Aldrich. ACM Trans. Program. Lang. Syst. (TOPLAS) 42, 3, Article 14, December 2020.
3. Gradual verification of recursive heap data structures. Jenna Wise, Johannes Bader, Cameron Wong, Jonathan Aldrich, Eric Tanter, and Joshua Sunshine. Proc. ACM Program. Lang. 4, OOPSLA, Article 228, November 2020
4. Can advanced type systems be usable? An empirical study of ownership, assets, and typestate in Obsidian. Michael Coblenz, Jonathan Aldrich, Joshua Sunshine, and Brad A. Myers. Proc. ACM Program. Lang. 4, OOPSLA, Article 132, November 2020
5. Penrose: from mathematical notation to beautiful diagrams. Katherine Ye, Wode Ni, Max Krieger, Dor Ma'ayan, Jenna Wise, Jonathan Aldrich, Joshua Sunshine, and Keenan Crane. ACM Trans. Graph., Vol. 39, No. 4, Article 144, 2020.
6. Decidable Subtyping for Path Dependent Types. Julian MacKay, Alex Potanin, Jonathan Aldrich, and Lindsay Groves. Proc. ACM Program. Lang. 4(POPL):66, 2020.
7. Model-Based Adaptation for Robotics Software. Jonathan Aldrich, David Garlan, Christian Kästner, Claire Le Goues, Anahita Mohseni-Kabir, Ivan Ruchkin, Selva Samuel, Bradley R. Schmerl, Christopher Steven Timperley, Manuela Veloso, Ian Voysey, Joydeep Biswas, Arjun Guha, Jarrett Holtz, Javier Cámara, Pooyan Jamshidi. IEEE Software 36.2 (83-90), 2019.
8. Reasonable Programmable Literal Notation. Cyrus Omar and Jonathan Aldrich. Proc. ACM Program. Lang. 2(ICFP):106, 2018.
9. Foundations of Typestate-Oriented Programming. Ronald Garcia, Eric Tanter, Roger Wolff, and Jonathan Aldrich. *Transactions on Programming Languages and Systems* 36(4) article 12, 2014.
10. AEminium: A Permission Based Concurrent-by-Default Programming Language Approach. Sven Stork, Karl Naden, Joshua Sunshine, Manuel Mohr, Alcides Fonseca, Paulo Marques, and Jonathan Aldrich. *Transactions on Programming Languages and Systems* 36(1) article 2, 2014.
11. A Case Study on the Lightweight Verification of a Multi-Task Threaded Task Server. Néstor Cataño, Ijaz Ahmed, Radu Siminiceanu, and Jonathan Aldrich. *Science of Computer Programming* 80(A):169-187, 2014.
12. Differencing and Merging of Architectural Views. Marwan Abi-Antoun, Jonathan Aldrich, Nagi Nahas, Bradley Schmerl, and David Garlan. *Automated Software Engineering Journal* 15(1):35-74, 2008.
13. A Case Study in Re-engineering to Enforce Architectural Control Flow and Data Sharing. Marwan Abi-Antoun, Jonathan Aldrich, and Wesley Coelho. *Journal of Systems and Software* 80(2):240-264, February 2007.
14. Discovering Architectures from Running Systems. Bradley Schmerl, Jonathan Aldrich, David Garlan, Rick Kazman, and Hong Yan. *IEEE Transactions on Software Engineering* 32(7):454-466, July 2006.
15. Comprehensive Synchronization Elimination for Java. Jonathan Aldrich, Emin Gun Sirer, Craig Chambers, and Susan Eggers. *Science of Computer Programming* 47(2-3):91-120, May-June 2003.

Refereed Conference Publications

1. Gradual Program Analysis for Null Pointers. Sam Estep, Jenna Wise, Jonathan Aldrich, Éric Tanter, Johannes Bader, and Joshua Sunshine. Proc. European Conference on Object-Oriented Programming (ECOOP), 2021.

2. Facilitating Connector Evolution With Architecture-Centric Development. Selva Samuel and Jonathan Aldrich. Short Technical Track Paper, Proc. International Conference on Software Architecture (ICSA), March 2021.
3. A Case Study in Language-Based Security: Building an I/O Library for Wyvern. Jennifer Fish, Darya Melicher, and Jonathan Aldrich. Onward! 2020.
4. Syntactically Restricting Bounded Polymorphism for Decidable Subtyping. Julian MacKay, Alex Potanin, Jonathan Aldrich, and Lindsay Groves. Asian Symposium on Programming Languages and Systems (APLAS), 2020.
5. Interdisciplinary Programming Language Design. Michael Coblenz, Jonathan Aldrich, Brad A. Myers, and Joshua Sunshine. In Onward! Essays, 2018.
6. Capabilities: Effects for Free. Aaron Craig, Alex Potanin, Lindsay Groves, and Jonathan Aldrich. Proc. International Conference on Formal Engineering Methods (ICFEM), 2018.
7. Gradual Program Verification. Johannes Bader, Jonathan Aldrich, and Eric Tanter. Proc. VMCAI, January 2018.
8. The Implementation of Object Propositions: the Oprop Verification Tool. Ligia Nistor and Jonathan Aldrich. Proc. Formal Aspects of Component Software (FACS), 2017.
9. A Capability-Based Module System for Authority Control. Darya Melicher, Yanqingwei Shi, Alex Potanin, and Jonathan Aldrich. Proc. European Conference on Object-Oriented Programming (ECOOP), 2017.
10. Glacier: Transitive Class Immutability for Java. Michael Coblenz, Whitney Nelson, Jonathan Aldrich, Brad Myers and Joshua Sunshine. Proc. International Conference on Software Engineering (ICSE), Buenos Aires, Argentina, May 20-28, 2017.
11. Toward Semantic Foundations for Program Editors. Cyrus Omar, Ian Voysey, Michael Hilton, Joshua Sunshine, Claire Le Goues, Jonathan Aldrich and Matthew A. Hammer. Proc. Summit on Advances in Programming Languages (SNAPL), May 2017.
12. Hazelnut: A Bidirectionally Typed Structure Editor Calculus. Cyrus Omar, Ian Voysey, Michael Hilton, Jonathan Aldrich, and Matthew A. Hammer. Proc. Principles of Programming Languages (POPL), 2017.
13. Programmable Semantic Fragments: The Design and Implementation of typy. Cyrus Omar and Jonathan Aldrich. Proc. Generative Programming: Concepts & Experiences (GPCE), 2016.
14. Software Development Practices, Barriers in the Field and the Relationship to Software Quality. Beth Yost, Michael Coblenz, Brad Myers, Joshua Sunshine, Jonathan Aldrich, Sam Weber, Forrest Shull, Matthew Patron, Melissa Heeren, Shelley Krueger, and Mark Pfaff. Proc. Empirical Software Engineering and Measurement (ESEM), 2016.
15. Composing Interfering Abstract Protocols. Filipe Militão, Jonathan Aldrich, and Luís Caires. Proc. European Conference on Object-Oriented Programming, 2016.
16. Exploring Language Support for Immutability. Michael Coblenz, Joshua Sunshine, Jonathan Aldrich, Brad Myers, Samuel Weber, and Forrest Shull. *Proc. International Conference on Software Engineering (ICSE)*, 2016.
17. Inter-app Communication in Android: Developer Challenges. Waqar Ahmad, Christian Kästner, Joshua Sunshine, and Jonathan Aldrich. *Proc. Mining Software Repositories (MSR)*, 2016.
18. Cooperative Exceptions for Concurrent Objects. Bruno Cabral, Alcides Fonseca, Paulo Marques, and Jonathan Aldrich. *Proc. 21st IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2015.
19. A Course-Based Usability Analysis of Cilk Plus and OpenMP. Michael Coblenz, Robert Seacord, Brad Myers, Joshua Sunshine, and Jonathan Aldrich. *Visual Languages and Human-Centric Computing (VL/HCC)*, 2015.

20. A Theory of Tagged Objects. Joseph Lee, Jonathan Aldrich, Troy Shaw, and Alex Potanin. *Proc. European Conference on Object-Oriented Programming (ECOOP)*, 2015.
21. Composable and Hygienic Typed Syntax Macros. Cyrus Omar, Chenglong Wang, and Jonathan Aldrich. *Proc. Symposium on Applied Computing (SAC)*, 2015.
22. Searching the State Space: A Qualitative Study of API Protocol Usability. Joshua Sunshine, James Herbsleb and Jonathan Aldrich. *Proc. International Conference on Program Comprehension (ICPC)*, 2015.
23. Collaborative Infrastructure for Test-Driven Scientific Model Validation. Cyrus Omar, Jonathan Aldrich, and Richard Gerkin. *Proc. International Conference on Software Engineering, New Ideas and Results track (ICSE NIER)*, 2014.
24. Structuring Documentation to Support State Search: A Laboratory Experiment about Protocol Programming. Joshua Sunshine, James Herbsleb, and Jonathan Aldrich. *Proc. European Conference on Object-Oriented Programming*, 2014.
25. Rely-Guarantee Protocols. Filipe Militão, Jonathan Aldrich, and Luís Caires. *Proc. European Conference on Object-Oriented Programming*, 2014.
26. Safely Composable Type-Specific Languages. Cyrus Omar, Darya Kurilova, Ligia Nistor, Benjamin Chung, Alex Potanin, and Jonathan Aldrich. *Proc. European Conference on Object-Oriented Programming*, 2014. **Distinguished Paper Award.**
27. In-Nimbo Sandboxing. Michael Maass, Jonathan Aldrich, and William Scherlis. *Proc. Science of Security (HotSOS)*, 2014.
28. Object Propositions. Ligia Nistor, Jonathan Aldrich, Stephanie Balzer and Hannes Mehnert. *Proc. Formal Methods*, 2014.
29. The Power of Interoperability: Why Objects Are Inevitable. Jonathan Aldrich. In *Onward! Essays*, 2013.
30. Introducing Tool-Supported Architecture Review into Software Design Education. Yuanfang Cai, Rick Kazman, Ciera Jaspan, and Jonathan Aldrich. Proc. *Software Engineering Education and Training (CSEE&T)*, 2013.
31. Statically Checking API Protocol Conformance with Mined Multi-Object Specifications. Michael Pradel, Ciera Jaspan, Jonathan Aldrich, and Thomas Gross. In *Proceedings of the International Conference on Software Engineering (ICSE '12)*, 2012.
32. Verification of Snapshotable Trees using Access Permissions and Typestate. Hannes Mehnert and Jonathan Aldrich. In *Proceedings of TOOLS*, 2012.
33. A Type System for Borrowing Permissions. Karl Naden, Robert Bocchino, Kevin Bierhoff, Jonathan Aldrich. In *Proceedings of Principles of Programming Languages (POPL '12)*, 2012.
34. First-Class State Change in Plaid. Joshua Sunshine, Karl Naden, Sven Stork, Jonathan Aldrich, and Eric Tanter. In *Proceedings of Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '11)*, 2011.
35. Gradual Typestate. Roger Wolff, Ronald Garcia, Eric Tanter, and Jonathan Aldrich. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP '11)*, 2011.
36. An Empirical Study of Object Protocols in the Wild. Nels E. Beckman, Duri Kim, and Jonathan Aldrich. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP '11)*, 2011.
37. Permission-Based Programming Languages (NIER Track). Jonathan Aldrich, Ronald Garcia, Mark Hahnenberg, Manuel Mohr, Karl Naden, Darpan Saini, Sven Stork, Joshua Sunshine, Eric Tanter, and Roger Wolff. In *Proceedings of the International Conference on Software Engineering (ICSE '11), New Ideas and Emerging Results Track*, 2011.

38. Static Extraction and Conformance Analysis of Hierarchical Runtime Architectural Structure using Annotations. Marwan Abi-Antoun and Jonathan Aldrich. In *Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '09)*, 2009.
39. CZ: Multiple Inheritance without Diamonds. Donna Malayeri and Jonathan Aldrich. *Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '09)*, 2009.
40. Typestate-Oriented Programming. Jonathan Aldrich, Joshua Sunshine, Darpan Saini, and Zachary Sparks. In *Proceedings of Onward!*, 2009.
41. Concurrency by Default: Using Permissions to Express Dataflow in Stateful Programs. Sven Stork, Paulo Marques, and Jonathan Aldrich. In *Proceedings of Onward!*, 2009.
42. Checking Framework Interactions with Relationships. Ciera Jaspan and Jonathan Aldrich. In *Proceedings of the European Conference on Object Oriented Programming (ECOOP '09)*, July 2009.
43. Practical API Protocol Checking with Access Permissions. Kevin Bierhoff, Nels E. Beckman, and Jonathan Aldrich. In *Proceedings of the European Conference on Object Oriented Programming (ECOOP '09)*, July 2009.
44. Is Structural Subtyping Useful? An Empirical Study. Donna Malayeri and Jonathan Aldrich. In *Proceedings of the European Symposium on Programming (ESOP '09)*, March 2009.
45. Verifying Correct Usage of Atomic Blocks and Typestate. Nels Beckman, Kevin Bierhoff, and Jonathan Aldrich. In *Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '08)*, Nashville, TN, USA, October 2008.
46. Error Reporting Logic. Ciera Jaspan, Trisha Quan, and Jonathan Aldrich. In *Proceedings of the International Conference on Automated Software Engineering (ASE '08)*, September 2008.
47. Integrating Nominal and Structural Subtyping. Donna Malayeri and Jonathan Aldrich. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP '08)*, July 2008.
48. Using Types to Enforce Architectural Structure. Jonathan Aldrich. In *Working International Conference on Software Architecture (WICSA '08)*, February 2008.
49. Modular Typestate Checking of Aliased Objects. Kevin Bierhoff and Jonathan Aldrich. In *Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '07)*, October 2007.
50. Checking Semantic Usage of Frameworks. Ciera Jaspan and Jonathan Aldrich. In *Library Centric Software Design Symposium*, 2007.
51. Differencing and Merging of Architectural Views. Marwan Abi-Antoun, Jonathan Aldrich, Nagi Nahas, Bradley Schmerl, and David Garlan. In *Proceedings of the International conference on Automated Software Engineering*, September 2006.
52. Lightweight Object Specification with Typestates. Kevin Bierhoff and Jonathan Aldrich. In *Proceedings of Foundations of Software Engineering (FSE '05)*, September 2005.
53. Permission-Based Ownership: Encapsulating State in Higher-Order Typed Languages. Neel Krishnaswami and Jonathan Aldrich. In *Proceedings of Programming Language Design and Implementation (PLDI '05)*, June 2005.
54. Open Modules: Modular Reasoning about Advice. Jonathan Aldrich. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP '05)*, July 2005.
55. Prototypes with Multiple Dispatch: An Expressive and Dynamic Object Model. Lee Salzman and Jonathan Aldrich. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP '05)*, July 2005.

56. Ownership Domains: Separating Aliasing Policy from Mechanism. Jonathan Aldrich and Craig Chambers. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP '04)*, June 2004.
57. DiscoTect: A System for Discovering Architectures from Running Systems. Hong Yan, David Garlan, Bradley Schmerl, Jonathan Aldrich, and Rick Kazman. In *Proceedings of International Conference on Software Engineering (ICSE '04)*, May 2004.
58. Language Support for Connector Abstractions. Jonathan Aldrich, Vibha Sazawal, Craig Chambers, and David Notkin. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP '03)*, July 2003.
59. Alias Annotations for Program Understanding. Jonathan Aldrich, Valentin Kostadinov, and Craig Chambers. In *Proceedings of Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '02)*, November 2002.
60. Architectural Reasoning in ArchJava. Jonathan Aldrich, Craig Chambers, and David Notkin. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP '02)*, June 2002.
61. ArchJava: Connecting Software Architecture to Implementation. Jonathan Aldrich, Craig Chambers, and David Notkin. In *Proceedings of the International Conference on Software Engineering (ICSE '02)*, May 2002.
62. Static Analyses for Eliminating Unnecessary Synchronization from Java Programs. Jonathan Aldrich, Craig Chambers, Emin Gun Sirer, and Susan Eggers. In *Proceedings of the Sixth International Static Analysis Symposium (SAS '99)*, September 1999.

Workshop Papers, Demonstrations, and Technical Reports

1. An Empirical Study of Protocols in Smart Contracts. Timothy Mou, Michael Coblenz, and Jonathan Aldrich. HATRA 2021.
2. Psamathe: A DSL with Flows for Safe Blockchain Assets. Reed Oei, Michael Coblenz, and Jonathan Aldrich. PADL 2021.
3. User-Cenetered Programming Language Design: A Course-Based Case Study. Michael Coblenz, Ariel Davis, Megan Hofmann, Vivian Huang, Siyue Jin, Max Krieger, Kyle Liang, Brian Wei, Mengchen Sam Yong, and Jonathan Aldrich. HATRA 2020.
4. Gradual Program Analysis. Sam Estep, Jenna Wise, Jonathan Aldrich, Eric Tanter, Johannes Bader, and Joshua Sunshine. In *Workshop on Gradual Typing (WGT'20)*, 2020.
5. Gradual Verification of Recursive Heap Data Structures. Jenna Wise, Johannes Bader, Jonathan Aldrich, Eric Tanter, and Joshua Sunshine. In *Workshop on Gradual Typing (WGT'20)*, 2020.
6. Smarter Smart Contract Development Tools. Michael Coblenz, Joshua Sunshine, Jonathan Aldrich, and Brad A. Myers. Proc *WETSEB*, 2019.
7. A Pilot Study of the Safety and Usability of the Obsidian Blockchain Programming Language. Gauri Kambhatla, Michael Coblenz, Reed Oei, Joshua Sunshine, Brad Myers, and Jonathan Aldrich. PLATEAU, 2019.
8. User-Centered Design of Permissions, Typestate, and Ownership in the Obsidian Blockchain Language. Michael Coblenz, Jonathan Aldrich, Joshua Sunshine, and Brad A. Myers. Proc. *HCI for Blockchain: Studying, Designing, Critiquing and Envisioning Distributed Ledger Technologies Workshop at CHI'2018*, 2018.
9. A User Study to Inform the Design of the Obsidian Blockchain DSL. Celeste Barnaby, Michael Coblenz, Tyler Etzel, Eliezer Kanal, Joshua Sunshine, Brad Myers, and Jonathan Aldrich. Proc. *Workshop on Evaluation and Usability of Programming Languages and Tools (PLATEAU'2017)*, 2017.

10. Empirical Studies on the Security and Usability Impact of Immutability. Sam Weber, Michael Coblenz, Brad Myers, Jonathan Aldrich, and Joshua Sunshine. Proc. *IEEE Cybersecurity Development Conference*, 2017.
11. Substance and Style: domain-specific languages for mathematical diagrams. Wode Ni*, Katherine Ye*, Joshua Sunshine, Jonathan Aldrich, and Keenan Crane. Proc. *Domain-Specific Language Design and Implementation (DSLDI)*, 2017.

    *indicates equal contribution.
12. Designing extensible, domain-specific languages for mathematical diagrams. Katherine Ye, Keenan Crane, Jonathan Aldrich, and Joshua Sunshine. In *Off the Beaten Track (OBT)*, 2017.
13. Capability Safe Reflection for the Wyvern Language. Esther Wang and Jonathan Aldrich. In *Proceedings of the Workshop on Meta-Programming Techniques and Reflection (META)*, 2016.
14. Naturally Embedded DSLs. Jonathan Aldrich and Alex Potanin. In *Proceedings of the Workshop on Domain-Specific Language Design and Implementation (DSLDI)*, 2016.
15. Delegation Revisited: Reuse Mechanisms in a Statically Typed, Expression-Oriented Language. Jonathan Aldrich and Alex Potanin. In *Proceedings of the Workshop on New Object-Oriented Languages (NOOL)*, 2016.
16. Delegation vs. Inheritance for Typestate Analysis. Du Li, Alex Potanin, and Jonathan Aldrich. In *Proceedings of Formal Techniques for Java Like Programs (FTfJP)*, 2015.
17. Statically Typed String Sanitation Inside a Python. Nathan Fulton, Cyrus Omar, and Jonathan Aldrich. Proc. *Privacy and Security in Programming (PSP)*, 2014. **Best Paper Award.**
18. Usability Hypotheses in the Design of Plaid. Jonathan Aldrich and Joshua Sunshine. In *Evaluation and Usability of Programming Languages and Tools (PLATEAU)*, 2014.
19. Wyvern: Impacting Software Security via Programming Language Design. Darya Kurilova, Alex Potanin, and Jonathan Aldrich. Proc. *Evaluation and Usability of Programming Languages and Tools (PLATEAU)*, 2014.
20. Considering Productivity Effects of Explicit Type Declarations. Michael Coblenz, Jonathan Aldrich, Brad Myers, and Joshua Sunshine. In *Evaluation and Usability of Programming Languages and Tools (PLATEAU)*, 2014.
21. Language-Based Architectural Control. Jonathan Aldrich, Cyrus Omar, Alex Potanin, and Du Li. In *International Workshop on Aliasing, Capabilities, and Ownership (IWACO '14)*, 2014.
22. Using Machine Learning in the Automatic Translation of Object Propositions. Ligia Nistor and Jonathan Aldrich. Proc. AI4FM, 2014.
23. Substructural Typestates. Filipe Militão, Jonathan Aldrich, and Luís Caires. In *Programming Languages meets Program Verification*, 2014.
24. Wyvern: A Simple, Typed, and Pure Object-Oriented Language. Ligia Nistor, Darya Kurilova, Stephanie Balzer, Benjamin Chung, Alex Potanin, and Jonathan Aldrich. In *Mechanisms for Specialization, Generalization, and Inheritance (MASPEGHI)*, 2013.
25. Type-Directed, Whitespace-Delimited Parsing for Embedded DSLs. Cyrus Omar, Benjamin Chung, Darya Kurilova, Alex Potanin, and Jonathan Aldrich. In *Globalization of Domain Specific Languages (GlobalDSL)*, 2013.
26. High-Level Abstractions for Safe Parallelism. Robert L. Bocchino, Hannes Mehnert, and Jonathan Aldrich. In *Workshop on Determinism and Correctness in Parallel Programming*, 2013.
27. Are Object Protocols Burdensome? Ciera Jaspan and Jonathan Aldrich. In Evaluation and Usability of Programming Languages and Tools Workshop (PLATEAU), 2011.

28. Verifying Object-Oriented Code Using Object Propositions. Ligia Nistor and Jonathan Aldrich. In *International Workshop on Aliasing, Confinement and Ownership in object-oriented programming (IWACO)*, 2011.
29. Featherweight Typestate. Ronald Garcia, Roger Wolff, Eric Tanter, and Jonathan Aldrich. *Technical Report CMU-ISR-10-110*, July 2010.
30. Gradual Featherweight Typestate. Roger Wolff, Ronald Garcia, Eric Tanter, and Jonathan Aldrich. *Technical Report CMU-ISR-10-116R*, July 2010 (revised December 2010).
31. Aliasing Control with View-Based Typestate. Filipe Militão, Jonathan Aldrich, and Luís Caires. In *Proceedings of Formal Techniques for Java Like Programs (FTfJP)*, 2010.
32. A Theory of Typestate-Oriented Programming. Darpan Saini, Joshua Sunshine, and Jonathan Aldrich. In *Proceedings of Formal Techniques for Java Like Programs (FTfJP)*, 2010.
33. DynXML: Safely Programming the Dynamic Web. Joshua Sunshine and Jonathan Aldrich. In *Proceedings of Analysis and Programming Languages for Web Applications and Cloud Applications (APLWACA)*, 2010.
34. Modular Composition and State Update in Plaid. Jonathan Aldrich, Karl Naden, and Eric Tanter. In *Proceedings of the Workshop on Mechanisms for Specialization, Generalization, and Inheritance (MASPEGHI)*, 2010.
35. Resource-Based Programming in Plaid. Jonathan Aldrich. Unpublished manuscript presented at the *Fun and Innovative Thoughts (FIT) session at PLDI*, 2010.
36. A Language-based Approach to Specification and Enforcement of Architectural Protocols. Kevin Bierhoff, Darpan Saini, Matthew Kehrt, Majid Al-Meshari, Sangjin Han, and Jonathan Aldrich. *Technical Report CMU-ISR-10-110*, March 2010.
37. Verifying Event-Driven Programs using Ramified Frame Properties. Neelakantan R. Krishnaswami, Jonathan Aldrich, and Lars Birkedal. In *Proceedings of Types in Language Design and Implementation (TLDI)*, 2010.
38. Typestate Protocol Specification in JML. Taekgoo Kim, Kevin Bierhoff, Jonathan Aldrich, and Sungwon Kang. In *Proceedings of the Workshop on Specification and Verification of Component-Based Systems (SAVCBS '09)*, August 2009.
39. Reducing STM Overhead with Access Permissions. Nels E. Beckman, Yoon Phil Kim, Sven Stork, and Jonathan Aldrich. In *Proceedings of the International Workshop on Aliasing, Confinement and Ownership 2009 (IWACO '09)*, July 2009.
40. Language support for Distributed Proxies. Darpan Saini, Joshua Sunshine, and Jonathan Aldrich. Proc. *Distributed Objects for the 21st Century*, 2009.
41. Static Extraction of Sound Hierarchical Runtime Object Graphs. Marwan Abi-Antoun and Jonathan Aldrich. In *Proceedings of Types in Language Design and Implementation*, January 2009.
42. Design Patterns in Separation Logic. Neelakantan R. Krishnaswami, Jonathan Aldrich, Lars Birkedal, Kasper Svendsen, and Alexandre Buisse. In *Proceedings of Types in Language Design and Implementation*, January 2009.
43. A Field Study in Static Extraction of Runtime Architectures. Marwan Abi-Antoun and Jonathan Aldrich. In *Proceedings of the Workshop on Program Analysis for Software Tools and Engineering (PASTE'08)*, November 2008.
44. Permissions to Specify the Composite Design Pattern. Kevin Bierhoff and Jonathan Aldrich. In *proceedings of the FSE 2008 Workshop on Specification and Verification of Component-Based Systems (SAVCBS '08)*, November 2008.
45. Verifying Correct Usage of Atomic Blocks and Typestate: Technical Companion. Nels Beckman and Jonathan Aldrich. *Carnegie Mellon University Technical Report CMU-ISR-08-126*, 2008.

46. SASyLF: An Educational Proof Assistant for Language Theory. Jonathan Aldrich, Robert J. Simmons, and Key Shin. In *Proceedings of Functional and Declarative Programming in Education (FDPE '08)*, 2008.
47. A Theory of Linear Objects. Matthew Kehrt and Jonathan Aldrich. In *2008 International Workshop on Foundations of Object-Oriented Languages (FOOL '08)*, San Francisco, California, January 2008.
48. Checking and Measuring the Architectural Structural Conformance of Object-Oriented Systems. Marwan Abi-Antoun and Jonathan Aldrich. *Carnegie Mellon University Technical Report CMU-ISRI-07-119*, December 2007.
49. Modular Verification of the Subject-Observer Pattern via Higher-Order Separation Logic. Neelakantan R. Krishnaswami, Lars Birkedal, and Jonathan Aldrich. In *ECOOP 2007 Workshop on Formal Techniques for Java Like Programs*, July 2007.
50. Ownership Domains in the Real World. Marwan Abi-Antoun and Jonathan Aldrich. In *International Workshop on Aliasing, Confinement and Ownership in object-oriented programming (IWACO)*, in conjunction with the European Conference on Object-Oriented Programming (ECOOP), 2007.
51. Compile-Time Views of Execution Structure Based on Ownership. Marwan Abi-Antoun and Jonathan Aldrich. In *International Workshop on Aliasing, Confinement and Ownership in object-oriented programming (IWACO)*, in conjunction with the European Conference on Object-Oriented Programming (ECOOP), 2007.
52. Eclipse Plug-ins for Statically Checking and Visualizing Ownership Domain Annotations. Marwan Abi-Antoun and Jonathan Aldrich. *Research Demonstration, European Conference on Object-Oriented Programming (ECOOP)*, 2007.
53. A Programming Model for Failure-Prone, Collaborative Robots. Nels Beckman and Jonathan Aldrich. In *the 2nd International Workshop on Software Development and Integration in Robotics (SDIR)*, Rome, Italy, April 2007.
54. Combining Structural Subtyping and External Dispatch. Donna Malayeri and Jonathan Aldrich. In *2007 International Workshop on Foundations and Development of Object-Oriented Languages (FOOL/WOOD'07)*, Nice, France, January 2007.
55. JavaD: Bringing Ownership Domains to Mainstream Java. Marwan Abi-Antoun and Jonathan Aldrich. *Carnegie Mellon University Technical Report CMU-ISRI-06-110*, May 2006.
56. A Language-based Approach to Specification and Enforcement of Architectural Protocols. Kevin Bierhoff, Jonathan Aldrich, and Sangjin Han. *Technical Report CMU-ISRI-06-121*, April 2006.
57. Ego: Controlling the Power of Simplicity. Andi Bejleri, Jonathan Aldrich, and Kevin Bierhoff. In *proceedings of the POPL '06 Workshop on Foundations of Object-Oriented Languages (FOOL '06)*, January 2006.
58. Improving System Dependability by Enforcing Architectural Intent. Marwan Abi-Antoun, Jonathan Aldrich, David Garlan, Bradley Schmerl, Nagi Nahas, and Tony Tseng. In *proceedings of the ICSE 2005 Workshop on Architecting Dependable Systems (WADS '05)*, May 2005.
59. Modeling and Implementing Software Architecture with Acme and ArchJava. Marwan Abi-Antoun, Jonathan Aldrich, David Garlan, Bradley Schmerl, Nagi Nahas, and Tony Tseng. *Demonstration, in proceedings of the International Conference on Software Engineering (ICSE '05)*, May 2005.
60. Selective Open Recursion: Modular Reasoning about Components and Inheritance. Jonathan Aldrich and Kevin Donnelly. In *proceedings of the FSE 2004 Workshop on Specification and Verification of Component-Based Systems (SAVCBS '04)*, November 2004.
61. Open Modules: Reconciling Extensibility and Modularity. Jonathan Aldrich. In *Proceedings of the Workshop on Software Engineering Properties of Languages for Aspect Technologies (SPLAT '04)*, March 2004.

62. Statically-Scoped Exceptions: a Typed Foundation for Aspect-Oriented Error Handling. Neel Krishnaswami and Jonathan Aldrich. *Carnegie Mellon Technical Report CMU-ISRI-05-102*, published on web January 2004, TR version January 2005.
63. Architecture-Centric Programming for Adaptive Systems. Jonathan Aldrich, Vibha Sazawal, Craig Chambers, and David Notkin. In *Proceedings of the Workshop on Self-Healing Systems (WOSS '02)*, November 2002.
64. Architecture-Centric Programming for Context-Aware Configuration. Vibha Sazawal and Jonathan Aldrich. In *Proceedings of the OOPSLA '02 Workshop on Engineering Context-Aware Object-Oriented Systems and Environments (ECOOSE '02)*, November 2002.
65. Challenge Problems for Separation of Concerns. Jonathan Aldrich. In *Proceedings of the OOPSLA 2000 Workshop on Advanced Separation of Concerns*, October 2000.
66. Evaluating Module Systems for Crosscutting Concerns. Jonathan Aldrich. *University of Washington PhD Generals Examination Report*, September 2000.
67. Providing Easier Access to Remote Objects in Client Server Systems. Jonathan Aldrich, James Dooley, Scott Mandelsohn, and Adam Rifkin. In *Thirty-first Hawaii International Conference on System Sciences (HICSS-31)*, January 1998.

Software Artifacts

Penrose. Open source software, available 2017-present. With Katherine Ye, Wode Ni, Max Krieger, Dor Ma'ayan, Yumeng Du, Stella Trout, Lily Shellhammer.

Obsidian compiler. Open source software, available 2018-present. With Michael Coblenz, Tyler Etzel, Gauri Kambhatla.

Wyvern interpreter. Open source software, available 2013-present. With Darya Melicher, Alex Potanin, Michael Plamann, Benjamin Chung, Amanda Liu, Robbie McKinstry, Aaron Craig, Valerie Zhao, Troy Shaw, Henry Nelson, Justin Lubin, Yangqingwei Shi, Abhinav, Chenglong Wang.

Wyvern interpreter. Open source software, available 2013-present. With Darya Melicher, Alex Potanin, Michael Plamann, Benjamin Chung, Amanda Liu, Robbie McKinstry, Aaron Craig, Valerie Zhao, Troy Shaw, Henry Nelson, Justin Lubin, Yangqingwei Shi, Abhinav, Chenglong Wang.

Plaid compiler. Open source software, available 2010-present. With Karl Naden, Joshua Sunshine, Sven Stork, and Mark Hahnenberg.

PLURAL typestate analysis tool. Open source software, available 2008-present. With Kevin Bierhoff and Nels Beckman.

SASyLF proof assistant. Open source software, available 2008-present. With Key Shin, Tye Wang, and Matthew Rodriguez.

Crystal static analysis infrastructure. Open source software, available 2008-present (previous internal development 2005-2008). With Nels Beckman, Kevin Bierhoff, Edwin Chan, David Dickey, Ciera Jaspan, Thomas LaToza, Dean Sutherland, and others.

The ArchJava Compiler and IDE. Open source software, available 2001-present.

Acme-ArchJava integration plugin. Open source software, available 2004-present.

ExnJava Eclipse plugin. Internal development, 2004-2005. With Donna Malayeri.

Dynamic typestate checking tool. Internal development, 2005. With Kevin Bierhoff.

Architecture synchronization tool. Available on request for research use. Developed 2004-present. With Marwan Abi-Antoun.

Architecture protocol checking tool. Internal development, 2006-present. With Kevin Bierhoff, Sangjin Han, Matthew Kehrt, Majid Al-Meshari, and Darpan Saini.

JavaD ownership system. Internal development, 2006-present. With Marwan Abi-Antoun.

Error reporting system. Internal development, 2006-present. With Ciera Jaspan and Trisha Quan.

Service

Conference Program Committees and Chairs

Onward! Steering Committee Chair, 2017-2020

OOPSLA/SPLASH Steering Committee Chair, 2017-2019

International Conference on Software Engineering (ICSE) PC 2019, 2015, 2005

SNAPL PC 2019

Object Oriented Programming Systems, Languages, and Applications (OOPSLA) PC 2023, 2018, 2016, 2012, 2008; ERC 2020

HotSoS PC 2016, 2018, 2019

OOPSLA 2017 Program Chair

APSEC PC 2016

SPLASH 2015 General Chair

Onward! Essays PC 2014

European Conference on Object-Oriented Programming (ECOOP) 2021 PC, 2014 ERC, 2011 PC, 2007 PC, and 2006 PC

Generative Programming: Concepts & Experiences (GPCE) PC 2013

International Conference on Objects, Models, Components, Patterns (TOOLS Europe) PC 2012

Conference on Aspect-Oriented Software Development (AOSD) 2012 and 2005 (PC), 2013 (Modularity Visions PC)

Programming Language Design and Implementation (PLDI) 2010 (PC), 2013 (ERC)

Brazilian Symposium on Programming Languages (SBLP) 2010 (program committee co-chair), 2011-2013 (program committee)

Working IEEE/IFIP Conference on Software Architecture (WICSA) PC 2009, 2008, and 2004

Joint Modular Languages Conference (JMLC) PC 2006

Other Service Outside Carnegie Mellon University

ACM SIG Governing Board Executive Committee Publications Advisor and ACM Publications Board member: July 2020-June 2022

Rising Starts workshop mentor, 2020

Long-term mentor, ICFP 2020 and SPLASH 2020

NSF panel service: 2020, 2017, 2016, and some prior years

2020,2021 ACM SIGPLAN John Reynolds Dissertation Award Committee member

PLACES 2020 PC

2016-present ACM SIGPLAN Programming Languages Achievement Award Committee member

Software, Programming, Languages, and Applications: Software for Humanity (SPLASH)
- Hybridization co-chair, 2021 and 2022
- Steering committee, 2014-present
- Tutorials chair, 2013
- Doctoral symposium chair, 2011

Editorial Board member, Journal of Object Technology, 2010-2018

Foundations of Software Engineering (FSE)
- Workshops chair, 2012
- Proceedings chair, 2010

Types in Language Design and Implementation (TLDI) 2012 program committee

International Workshop on Foundations of Object-Oriented Languages (FOOL)
- General chair, 2010-2011
- Steering committee, 2012-2013
- Program chair, 2009
- Program committee, 2008, 2012

Foundations of Coordination Languages and Software Architectures (FOCLASA) 2011 program committee

Workshop on Specification and Verification of Component-Based Systems (SAVCBS)
- Steering committee, 2007-2010
- Program chair, 2006
- Program committee, 2004, 2005, and 2007

International Workshop on Aliasing, Confinement and Ownership in object-oriented programming (IWACO): program committee, 2003 and 2007

Workshop on Foundations of Aspect-Oriented Languages (FOAL): program committee, 2004, 2005, and 2006

Foundations of Software Engineering (FSE): Posters program committee, 2006

International Conference on Software Engineering (ICSE): Demonstrations program committee, 2005

Carnegie Mellon University Service

2020-present	Academic Freedom Commission
2020-present	SCS Undergraduate Review Committee
2020, 2021	ISR Representative, SCS Council
2020	ISR Hiring Committee
2004-present	SE Ph.D. admissions committee
2012-present	Member, Computer Science Department Speaker's Club
2014-2019	Director, ISR Software Engineering Ph.D. Program
2015-2017	Diversity Liaison, Software Engineering Faculty Search Committee
2014	Created the first edition of ISR's REUSE summer research program
2007-2014	Director, ISR Software Engineering Undergraduate Minor Program
2008-2012	Chair, Software Engineering Faculty Search Committee
2009-2011	Fellowship Nomination Committee
2009	SCS Dean Review Committee

2006	Graduate Student Retention Workgroup
2005-2007	Chair, ISR Undergraduate Software Engineering committee
2004	CSD admissions committee

Teaching

Fall 2021. Instructor, 17-363/17-663 (Programming Language Pragmatics), Carnegie Mellon University.

Spring 2021. Co-instructor, 17-355/17-665/17-819 (Program Analysis), Carnegie Mellon University.

Spring 2021. Co-instructor, 15-400 (Research Practicum in Computer Science), Carnegie Mellon University.

Fall 2020. Co-instructor, 15-300 (Research and Innovation in Computer Science), Carnegie Mellon University.

Spring 2020. Instructor, 17-396/17-696/17-960 (Language Design and Prototyping), Carnegie Mellon University.

Spring 2020. Co-instructor, 15-400 (Research Practicum in Computer Science), Carnegie Mellon University.

Fall 2019. Co-instructor, 15-300 (Research and Innovation in Computer Science), Carnegie Mellon University.

Spring 2019. Co-instructor, 15-400 (Research Practicum in Computer Science), Carnegie Mellon University.

Spring 2019. Instructor, 17-355/17-665/17-819 (Program Analysis), Carnegie Mellon University.

Fall 2018. Co-instructor, 15-300 (Research and Innovation in Computer Science), Carnegie Mellon University.

Spring 2018. Co-instructor, 15-400 (Research Practicum in Computer Science), Carnegie Mellon University.

Spring 2018. Co-instructor, 17-355/17-665/17-819 (Program Analysis), Carnegie Mellon University.

Fall 2017. Co-instructor, 15-300 (Research and Innovation in Computer Science), Carnegie Mellon University.

Spring 2017. Co-instructor, 15-400 (Research Practicum in Computer Science), Carnegie Mellon University.

Spring 2017. Instructor, 17-355/17-665 (Program Analysis), Carnegie Mellon University.

> Adapted the 15-819O graduate course on program analysis (below) for an undergraduate audience.

Fall 2016. Co-instructor, 15-300 (Research and Innovation in Computer Science), Carnegie Mellon University.

Spring 2016. Co-instructor, 15-400 (Research Practicum in Computer Science), Carnegie Mellon University.

> Co-developed 2-course sequence introducing undergraduate students to research. Todd Mowry taught the first course in the sequence and we are co-teaching the second course; 12 students have continued into this second course, which is modest but in line with our first year goals, and this has led to an overall increase in research participation among our undergraduate students.

Fall 2015. Co-instructor, 15-214 (Principles of Software Construction: Objects, Design, and Concurrency), Carnegie Mellon University.

Fall 2014. Co-instructor, 15-214 (Principles of Software Construction: Objects, Design, and Concurrency), Carnegie Mellon University.

Spring 2014. Co-instructor, 15-413 (Software Engineering Practicum), Carnegie Mellon University.

Fall 2013. Co-instructor, 15-214 (Principles of Software Construction: Objects, Design, and Concurrency), Carnegie Mellon University.

Spring 2013. Instructor, 15-819O (Program Analysis), Carnegie Mellon University.

Fall 2012. Co-instructor, 15-214 (Principles of Software Construction: Objects, Design, and Concurrency), Carnegie Mellon University.

Spring 2011. Co-instructor, 15-214 (Principles of Software Construction: Objects, Design, and Concurrency), Carnegie Mellon University.

Fall 2011. Instructor, 15-214 (Principles of Software System Construction), Carnegie Mellon University.

Spring 2011. Co-instructor, 17-654/17-754 (Analysis of Software Artifacts), Carnegie Mellon University.

Spring 2011. Instructor, 15-413 (Software Engineering Practicum), Carnegie Mellon University.

Fall 2010. Co-instructor, 15-313 (Foundations of Software Engineering), Carnegie Mellon University.

Fall 2010. Instructor, 17-413 (Software Engineering Reflection), Carnegie Mellon University.

Spring 2010. Instructor, 15-819M (Program Analysis), Carnegie Mellon University.

Spring 2010. Instructor, 15-413 (Software Engineering Practicum), Carnegie Mellon University.

Fall 2009. Co-instructor, 15-313 (Foundations of Software Engineering), Carnegie Mellon University.

Fall 2009. Instructor, 17-413 (Software Engineering Reflection), Carnegie Mellon University.

Spring 2009. Instructor, 17-654/17-754 (Analysis of Software Artifacts), Carnegie Mellon University.

Fall 2008. Instructor, 15-413 (Software Engineering Practicum), Carnegie Mellon University.

Fall 2008. Instructor, 17-413 (Software Engineering Reflection), Carnegie Mellon University.

Fall 2008. Co-instructor, 17-732 (Emerging Programming Paradigms), Carnegie Mellon University.

Spring 2008. Co-instructor, 15-313 (Foundations of Software Engineering), Carnegie Mellon University.

Spring 2008. Instructor, 17-654/17-754 (Analysis of Software Artifacts), Carnegie Mellon University.

Spring 2007. Co-instructor, 15-313 (Foundations of Software Engineering), Carnegie Mellon University.

Spring 2007. Instructor, 17-654/17-754 (Analysis of Software Artifacts), Carnegie Mellon University.

Spring 2006. Instructor, 17-654/17-754 (Analysis of Software Artifacts), Carnegie Mellon University. Refined course and recorded for distance education. 40 students in class.

Fall 2005. Instructor, 15-413 (Introduction to Software Engineering), Carnegie Mellon University. Redeveloped course to combine combine a strong technical focus with a capstone project providing the opportunity to practice engineering knowledge, skills, and practices in a realistic development setting with a real client. 18 students in class.

Spring 2005. Instructor, 17-654/17-754 (Analysis of Software Artifacts), Carnegie Mellon University. Redeveloped course to focus on a broad range of static and dynamic analysis techniques for programs and other artifacts in the software process. Extended course to fulfill software engineering Ph.D. star course requirement in Analysis. 40 students in class.

Spring 2005. Co-instructor, 17-898 (Software engineering graduate reading seminar - Modeling Dynamic Software Architectures), Carnegie Mellon University.

Fall 2004. Instructor, 15-819 (Programming languages graduate reading seminar - Objects and Aspects: Language Support for Extensible and Evolvable Software), Carnegie Mellon University.

Fall 2003. Co-instructor, 15-312 (Undergraduate Programming Languages), Carnegie Mellon University.

Developed new material on object-oriented programming languages.

Winter 2002. Teaching Assistant, CSE 503 (Graduate Software Engineering), University of Washington.

Winter 2001. Teaching Assistant, CSE 501 (Graduate Compilers), University of Washington.

Summer 1999. Pre-Doctoral Lecturer, CSE 143 (Computer Programming II), University of Washington.

Spring 1999. Teaching Assistant, CSE 143 (Computer Programming II), University of Washington.

Fall 1998. Teaching Assistant, CSE 505 (Graduate Programming Languages). University of Washington.

Advising

Ph.D. Advisees

Donna Malayeri (Ph.D. 2009, Google)
Kevin Bierhoff (Ph.D. 2009, Google)
Marwan Abi-Antoun (Ph.D. 2010, The Mathworks)
Nels Beckman (Ph.D. 2010, Google)
Ciera Jaspan (Ph.D. 2011, Google)
Neel Krishnaswami (Ph.D. 2012, co-advised with John Reynolds, lecturer (assistant professor equivalent) at the University of Cambridge)
Thomas LaToza (Ph.D. 2012, co-advised with Brad Myers, assistant professor at George Mason University)
Sven Stork (Ph.D. 2013, coadvised with Paulo Marques, Duolingo)
Josh Sunshine (Ph.D. 2014, systems scientist at CMU)
Filipe Militão (Ph.D. 2015, coadvised with Luís Caires, BBC)
Michael Maass (Ph.D. 2016, co-advised with William Scherlis, Aptiv)
Cyrus Omar (Ph.D. 2017, Assistant Professor at the University of Michigan)
Ligia Nistor (Ph.D. 2017, Oracle)
Darya Melicher (Ph.D. 2019, Google)
Michael Coblenz (Ph.D. 2020, co-advised with Brad Myers, postdoc at the University of Maryland)
Selva Samuel
Katherine Ye (co-advised with Keenan Crane)
Jenna Wise (co-advised with Joshua Sunshine)
Kyle Liang
Luis Gomes (co-advised with Vincent Hellendoorn and Rui Abreu)
Ian McCormack (co-advised with Joshua Sunshine)

Postdoctoral Advisees

Robert L. Bocchino (JPL)
Stephanie Balzer (CMU)
Du Li (HP Labs)

Master's Thesis Advisees

Anlun Xu. Extending Abstract Effects with Bounds and Algebraic Handlers, 2020.
Yu Xiang "Billy" Zhu. Nominal Wyvern: Employing Semantic Separation for Usability, 2019.
Johannes Bader. Gradual Program Verification with Implicit Dynamic Frames, 2016.

Manuel Mohr. Æminium Compilation Theory and Run-Time Implementation, 2011.
Duri Kim. An Empirical Study on the Frequency and Classification of Object Protocols in Java, 2009.
Taekgoo Kim. Towards Specification and Verification of Usage Protocol Using Typestates in JML, 2009.
GwanPyo Do. Reachable Reference Algorithm for Inferring Ownership Types in Object Oriented Programming Languages, 2008.
Yoon-Phil Kim. Permission-based Optimization for Efficient Software Transactional Memory, 2008.

Undergraduate Thesis Advisees

Esther Wang. Designing Capability Safe Reflection for the Wyvern Language, 2016.
Sarah Chasins. Undergraduate thesis advisee in CS (at Swarthmore). Completed 2012.
Mark Hahnenberg. Undergraduate thesis advisee in CS. Completed 2011.
Sneha Popley. Undergraduate thesis advisee in CS (at Texas Christian University). Completed 2010.
Key Shin. Undergraduate thesis advisee in CS. Completed 2007.
Matthew Kehrt. Undergraduate thesis advisee in CS. Completed 2006.
Will Cooper. Undergraduate thesis advisee in CS. Completed 2006.
Andi Bejleri. Undergraduate exchange student thesis advisee. Completed 2005.
Lee Salzman. Undergraduate thesis advisee in Logic and Computation. Completed 2004.

Masters Independent Study/Practicum Advisees (year completed)

Fuyao Zhou (2010-2012)
Aparup Banerjee (2010-2011)
Darpan Saini (2008-2010)
Kelvin Lim (2008)
Jeffrey Beckett (2008)
Majid Al-Meshari (2007)
Tim Kral (2007)
Joseph Ayo Akinyele (2007)
Lutz Wrage (independent study 2006)
Varun Dutt (2006)
Monica Page (2006)
Sangjin Han (2006)
Bhavana Rehani (2006)
David Dickey (2006)
Min Chen (2005)
Soumya Simanta (2005)
Prasanth Ramanand (2005)
Michael German (2005)
Animesh Kejriwal (2005)
Ben Madore (2005)

Undergraduate Independent Study Advisees (year completed)

Chris Martens (2008)
Matthew Rodriguez (2008, 2010)
Trisha Quan (2007)
Kevin McInerney (2007)
Tye Wang (2007)

Funding

2021 Ethereum Foundation grant, $42,000.

2020 Ethereum Foundation grant, $64,690.

2019 National Science Foundation award, "SHF: Medium: Gradual Verification." $1,017,511 over 4 years, 2019-2023. With Joshua Sunshine.

2019 National Science Foundation award, "SHF: Small: Declaratively Creating Semantics-driven Visualizations." $449,680 over 3 years, 2019-2022. With Joshua Sunshine and Keenan Crane.

2018 Facebook Testing and Verification Award, "Incremental Verification, Gradually." $50,000 gift. With Eric Tanter and Joshua Sunshine.

2017 National Security Agency Lablet, Co-PI (with William Scherlis as Lead PI). Renewable for 3 years.

2015 DARPA BRASS award, Lead PI (with 4 Co-PIs), "Intelligent Model-Based Adaptation for Mobile Robotics", $7.8 million over 4 years, November 2015-2019

2014 Stevens Institute, joint project with Christian Kästner and Joshua Sunshine, as part of a multi-faculty effort with William Scherlis as lead.

2014 National Security Agency Lablet, Co-PI (with William Scherlis as Lead PI). 2014-2017 ($2.3M in 2014).

2013 National Security Agency Lablet award, "Race Vulnerability Study and Hybrid Race Detection." $139,598 over 2 years, 2013-2014.

2012 National Security Agency Lablet award, "A Language and Framework for Development of Secure Mobile Applications." $500,000+ over 3 years, 2012-2014.

2012 National Science Foundation award, "Collaborative Research: Teaching Software Modularity through Architectural Review." $100,000 over 2 years, 2012-2014.

2011 National Science Foundation award, "SHF:Small:Foundations of Permission-Based Object-Oriented Languages." $500,000 over 3 years, 2011-2014.

2010 CMU|Portugal research award, "Aeminium: Freeing Programmers from the Shackles of Sequentiality." $321,0185 over 3 years, 2010-2012.

2008 National Science Foundation award, "CPA-SEL: Practical Typestate Verification with Assume-Guarantee Reasoning." $300,000 over 3 years, 2008-2010.

DARPA Computer Science Study Group member. $600,000 over 3 years, potentially renewable for 2 more years. 2007-2010.

2006 National Science Foundation CAREER award, "Lightweight Modeling and Enforcement of Architectural Behavior," $450,000 over 5 years. 2006-2010.

Human and Robotic Technology: Dependable Real-Time and Embedded Space Software, NASA (Michael Shafto, Program Manager), $1.1 million over one year (2005-2006). Project Lead.

ITR: Synthetic Reality: Physically Rendering Dynamic 3D Objects from Programmable Matter. National Science Foundation (Helen Gill, Progarm Manager), $662,000 over 2 years. Senior Personnel.

Integrating Software Architecture and Software Development. National Science Foundation (Sol Greenspan, Program Director), $300,000 over 3 years. Co-written with advisor Craig Chambers while a graduate student; portions subcontracted to my research group at CMU after graduation.

External Talks

"On Understanding Data Abstraction, Revisited," Revisited: William Cook's Impact on Our Understanding of Objects. Talk at the IFIP Working Group on Programming Language Design, 2022.

Gradual Verification. Talks at the IFIP Working Group on Programming Language Design, and at the IFIP Working Group on Software Implementation Technology, 2021.

Using Capabilities to Enforce Architectural and Security Constraints. Tlak at the IFIP Working Group on Software Implementation Technology, 2021.

Penrose: From Mathematical Notation to Beautiful Diagrams. Keynote at IEEE CCWC 2021.

Usability Evaluation of the Obsidian Smart Contract Language. Talk at the IFIP Working Group on Programming Language Design, 2021.

Obsidian: A Language for Smart Contracts Designed for Safety and Usability. Talks at the University of Zurich, the University of Lugano, the Science of Security Lablet meeting, and the CyLab Partners meeting, 2019.

Penrose: From Mathematical Notation to Beautiful Diagrams. Talks at the IFIP Working Group on Programming Language Design and at SNAPL, 2019.

How to PL Researchers Think? and Evidence about Software Engineers for PL Design. Talk at Dagstuhl Seminar 18061 on Evidence About Programmers for Programming Language Design, 2018.

Inderdisciplinary Programming Language Design . Talk in the Software Research Seminar (SSSG) at Carnegie Mellon University, 2018.

Usable Architecture-Based Security in the Wyvern Language. CyLab Distinguished Seminar, 2018.

Obsidian: Safely Programming Contracts on the Blockchain. Talk at the CyLab Partners Conference, 2017.

Glacier: Usable Enforcement of Transitive Immutability. Talk at the University of British Columbia, 2017.

On Science, Mathematics, and Engineering in Language Design: The Case of an Immutability Type System. Talk at the IFIP Working Group on Programming Language Design, 2016.

Using Capability-Based Modules to Enforce Secure Resource Usage. Talk given at the University of British Columbia and the University of Washington, 2016.

Capability-Based Architectural Control. Talk at the NSA Lablet Quarterly Meeting, 2016.

Achieving Architectural Control via Language Support for Capabilities. Talk at the IFIP Working Group on Programming Language Design, 2016.

Building Security In with Capability-Based Architectural Control. Talk at the CyLab Partners Conference, 2015.

Evaluating Language Designs. Talk at the IFIP Working Group on Programming Language Design, 2015.

Wyvern: An Extensible Language for Secure Systems. Seminars at the University of Chile and the University of Buenos Aires, 2015.

Wyvern: The Human Dimension of Programming Language Security. Talk at the "Reprogramming Programming" TTI/Vanguard conference, 2014.

Wyvern: Simplicity, Extensibility, and Architectural Control. Talk at the IFIP Working Group on Programming Language Design, 2014.

Wyvern: Coupling Security and Productivity in Language Design. Oracle Labs, 2013.

Connecting Software Architecture to Implementation: The Next 10 Years. Most Influential Paper of ICSE 2002 Award Talk, given at ICSE 2012. With Craig Chambers and David Notkin.

Designed-In Security for Mobile Applications. Invited talk at the Designed-In Security session at the High Confidence Software and Systems conference, 2012.

Permission-Based Programming in Plaid. Given as a invited guest at the IFIP 1.16 Working Group on Language Design, 2012.

Plural and Plaid: Protocols in Practice. Workshop on Behavioral Types, April 2011.

Plaid: A Permission-Based Programming Language. Seminar at Microsoft Research in Redmond, November 2010.

Access Permissions for Correct and Ubiquitous Concurrency. Seminar at Los Alamos National Laboratory, November 2010.

Plaid: Centering a Language Design around Resources. Keynote at the Brazilian Symposium on Programming Languages, September 2010.

Pragmatic Typestate Verification with Permissions. Seminars at Stanford University and the Massachusetts Institute of Technology, January-February 2010.

Protocol Verification for Aliased Objects using Invariant-Carrying Permissions. Seminar at the Unversity of Washington, October 2009.

Design Intent: A Principled Approach to Application Security. Invited talk at the Boeing Application Security Developer's Forum, September 2009.

Invariant-Carrying Permissions: Modular Dependence on Shared State. Dagstuhl Seminar, July 2009.

Typestate Verification for Aliased Objects using Assume-Guarantee Permissions. Seminar at the University Nova de Lisboa, July 2009.

Typestate Verification for Aliased Objects using Assume-Guarantee Permissions. Seminar at Portland State University, August 2008.

Define, don't Confine. Invited keynote for the 2008 Intercontinental Workshop on Aliasing, Confinement, and Ownership (IWACO '08), July 2008.

Typestate Verification for Aliased Objects using Assume-Guarantee Permissions. Seminar at Cornell University, June 2008.

Assuring Object-Oriented Architecture. Talk at Drexel University, April 2008.

Modular, Pragmatic Typestate Verification using Assume-Guarantee Permissions. Talk at University of British Columbia, February 2008.

Assuring Object-Oriented Architecture. Dahl-Nygaard Junior Prize Keynote Address, ECOOP 2007, August 2007.

Young Guns/OO: The Next Generation. Panel participant at OOPSLA 2006, October 2006.

Prospects for Software Assurance. Lockheed Martin, August 2006.

Tales from Dissertationland and the Job Hunt. Invited keynote at the ECOOP 2005 Doctoral Symposium, July 2005.

Aspects and Modularity: The Hope and the Challenge. Invited talk, panel on modularity at the AOSD '05 Workshop on Foundations of Aspect Languages, March 2005.

Engineering More Dependable Space Software. Talk at NASA Exploration Systems Technology & Systems Integration Technical Interchange Meeting, December 2004.

Using Types to Enforce Architectural Design. Invited talk, California Institute of Technology, October 2003.

Using Types to Enforce Architectural Design. Invited talk, University of Southern California, October 2003.

Consulting and Industry

Co-founder and CTO, Noteful LLC, 2022

Summer Teaching, Torhea Education Group, 2019

Expert witness in a smartphone patent lawsuit, 2011-2012

Architecture consultant for an embedded software system, 2006

Professional Societies

Senior Member, Association for Computing Machinery, and SIGSOFT and SIGPLAN special interest groups

Senior Member, Institute of Electrical and Electronics Engineers