Jan Vitek

Curriculum Vitae

j.vitek@neu.edu
Phone: (617) 749 8148
Nationality: USA + Swiss + Czech
Date of Birth: 6.9.66

June 10, 2024

Northeastern University
360 Huntington Ave.
Boston, MA 02115

Research Interests

Programming languages, Compilers, Software engineering, Data Science.

Employment

Professor Northeastern University.	7/14 –
Professor Charles University.	7/22 –
Software Architect Lacework	9/21 – 3/23
Professor Czech Technical University,	10/20 – 9/23
Professor Purdue University.	8/10 – 7/14
Chief Scientist Fiji Systems LLC, Indianapolis.	6/09 – 6/17
Visiting Professor Stanford University.	6/12 – 6/13
Visiting Researcher Oracle Labs, Redwood shores.	9/12 – 7/13
Scientific Advisor 0xData, Mountainview.	1/12 – 6/13
Academic Visitor IBM T.J Watson.	9/06 – 5/11
Associate Professor Purdue University.	8/05 – 8/10
Visiting Professor INRIA, Rocquencourt.	7/08 – 9/08
Visiting Professor EPFL.	1/06 – 7/06
Assistant Professor Purdue University.	8/99 – 7/05
Research Assistant University of Geneva,	9/94 – 7/99 9/89 – 8/90
Software Consultant International Labor Organization.	8/92 – 8/93
Research Assistant University of Victoria, CA	9/90 – 7/92

Education

University of Geneva 99
PhD in Information Systems
"The Seal Calculus – A calculus of mobile computations" Advisor: D. Tsichritzis

University of Victoria 95
MSc in Computer Science
"Compact Dispatch Tables for Dynamically Typed Languages" Advisor: R. N. Horspool

Awards and Honors

20 **Dahl-Nygaard Senior Prize**
19 **SIGPLAN Distinguished Service Award**
18 **ECOOP Test of Time Award**
18 **ISSTA Distinguished Artifact Award**
17 **OOPSLA Distinguished Artifact Award**
11 **Purdue University Faculty Scholar**
11 **Microsoft SEIF Research Award**
06 **IBM Faculty Award**
01 **NSF CAREER Award**

Positions in Scholarly Associations

Editor in Chief of the ACM Transactiob On Programming Languages, 2021–

Vice President of the Association Internationale pour les Technologies Objets, elected 2010–2018.

Past-Chair of ACM Special Interest Group on Programming Languages (**SIGPLAN**), 2015–2018.

Chair of the ACM Special Interest Group on Programming Languages (**SIGPLAN**), elected, 2012–2015.

Vice President of IFIP Working Group 2.4, elected, 2011–2013.

Editor in Chief of the Journal of Object Technology, 2013–2014.

Member of JSR 302 expert group Safety Critical Java, 2007–2012.

Member Scientific Committee of Ecole des Mines de Nantes, 2012–2016.

Member CominLabs International Advisory Committee, 2011–2016.

Steering Committees

SC Chair of SPLASH, 2022 – .

SC Member of SPLASH, 2012 – .

SC Chair of ISMM, 2010 – 2011.

SC Member of PLDI, 2011 – 2017.

SC Member of POPL, 2012 – 2017.

SC Member VEE 2005 – 2010.

SC Chair of PLDI, 2013 – 2015.

SC Member of ICFP, 2012 – 2016.

SC Member of COORDINATION 2007 – 2010.

SC Member of TOOLS Europe, 2010 – 2011.

Founding SC Member TRANSACT workshop, 2005.

Founding SC Member of International Summer School on Trends in Concurrency, 2006.

SC Member of the Java Technologies for Real-time and Embedded Systems workshop, 2005.

Research highlights

My research focuses on increasing the level of abstraction of our interface to the machines that carry out computation on our behalf. I am interested in programming languages in all their forms.

Confinement and Ownership. Back in 1998, James Noble, John Potter and myself were convinced that the spaghetti structure of the heap was a serious challenge to reasoning about object-oriented programs. The combination of aliasing, mutability and subtyping were particularly tricky to deal with. We proposed a mechanism for alias protection [130] which was later renamed to *Ownership Types*.[1] One challenge of the early work was the rather copious amount of annotations that had to be provided by programmers. To alleviate this and codify software engineering practices familiar to programmers, we proposed *Confined Types* [46]. With confined types, the number of annotations required was drastically reduced. Furthermore, confinement is sound [37] and can be inferred [34]. It also informed the development of the region types used for Real-time Java programs in [30]. In that work, implicit type annotations were used to determine where object would be allocated and their lifetime. The StreamFlex stream programming system [28] relied on a very similar notion for memory managment. We also used similar ideas for enforcing thread locality [90]. Another variant of ownership types was part of a version of the Safety Critical Java standard [149].

Dynamic languages. In collaboration with IBM research, I designed Thorn [88], a programming language that allows programmers to evolve scripts into robust programs [155]. One of Thorn's innovations is a type system based on *Like Types* [85]. Like types are type annotations that can be added gradually; they are the first gradual type system free of pathological performance degradation while still able to provide some guarantees. Empirical evaluation demonstrated that like types can be used by an optmizing compiler to speedup annotated programs (and can be adapted to a language such as TypeScript) [66]. Thorn motivated me to study the use, in the wild, of dynamic languages. For this JavaScript was a perfect playground [156]. We analyzed thousands of JavaScript web pages [82] yielding unique insights into programmers' use of features such as reflection [80]. This led to the development of JSBench [79], a tool for transforming web sites into benchmark. JSBench resulted from an unusual collaboration between Mozilla and Microsoft and was eventually adopted by Apple as a browser performance benchmark. We also looked at techniques for automatically inferring the behavior of calls to `eval` and replacing those calls by safer code [75].

Scalable Data Analytics. One of my long term interests is to help scientist with the analysis of complex data sets. This can be done with domain specific languages that support runtime code generation, as show with Terra [70], or with established languages like R. The R programming language is a widely used vehicle for statistical computating which has serious limitations in its ability to handle larger data sets. We started by trying to understand R, for this we formalized a small subset of the language and, in parallel, analyzed

[1] The term "ownership types" appears in over 4,000 papers according to Google Scholar attesting to the enduring popularity of the idea; a language like Rust is an example of practical adoption of the ideas.

statically and dynamically a body of 4 million lines of code [73]. The results of that effort motivated the development of FastR [67], an optimizing virtual machine for the R language which was adopted by Oracle research as the basis of their effort to integrate R with the Oracle database.

Language Implementation. I started my career with algorithms for speeding up some of the most frequent operations performed by object oriented programs, namely, method dispatch and type tests. I proposed a compact solution to the dispatching problem in dynamically typed languages in [136] and later reduced space requirements in [134] and looked architectural impacts of these solutions [135]. Later, I revisited dispatching with proposals for multiple dispatch policies [106], [31]. For type tests, the challenge was to find data structures optimized for space and time. We tried compact bitset encodings based on graph coloring [132], and then proposed an alternative that allowed constant time test [131] and finally devised an encoding that could be easily recomputed on loading of new clases [119]. These techniques were used in the Fiji Java virtual machine [154], [84]. I also led the Ovm project which delivered an open source framework for building language runtimes. By design, Ovm can be specialized and assembled into a configuration customized for a particular problem domain. Ovm was used in the first Real-Time Java virtual machine to be deployed and flight tested on a Boeing-built UAV [33].

Real-Time and Embedded Computing. Part of the Ovm project involved trying to demonstate that the Java language could be used for real-time computing. One obvious challenge was the memory management (or garbage collection) subsystem. My group produced Schism [81], the most efficient real-time garbage collection algortihm [87] in use in a commercial product [84]. We investigated real-time memory management techniques in a number of contexts [100], [96], [93], [29]. For real-time stream processing we achieved [28] periods of 50 μs without losing the portability or the memory safety of Java. StreamFlex offers a dataflow programming model with zero copy [95] and it makes an interesting use of software transactional memory [105] for communication with non-real-time tasks. The technology was transfered to IBM [92]. An ongoing project aims to formalize the guarantees needed for safety critical applications in Java [157]. Part of this work is being done in the context of the JSR-302 Safety Critical Java expert group. An early result is the development of new memory model for Java suitable for proving the correctness of compiler optimizations [71].

Concurrency control. Our empirical study of the DACAPO benchmark suite [74] demonstrated the limits of parallel execution in Java. New abstractions are needed to assist programmers. Our work on *atomic sets* allows concurrency control to be synthesized from high-level specificaitions that are part of the data declarations [83]. Atomic set leverage ideas from ownership types and confinement to decrease annotation burden, most use-cases require only a handful of annotations [25] and deadlocks can be prevented by program analysis [69]. I also investigated transactional memory abstractions: giving semantics to software transactional memory in [39], tuning the garbage collector to be transaction-aware in [147], and providing the first non-trivial benchmark suite [99] for transactional memory. Looking at the problem of ensuring predictable performance in hard real-time environments, I came up with *preemptible atomic regions* [105] which were later adopted in StreamFlex [91]. In another project, we looked at architectural support for real-time transactions [24].

Software Security. Ideas from the software transactional memory work came together with dynamic ownership tracking in our work on security for JavaScript based on delimited histories [68]. I investigated security off and on, with forays into intrusion detection techniques for C programs based on inlining automata [108], non-interference for a concurrent language, the box-π calculus, a minimal extension of the π-calculus with encapsulation [44], distributed access control [116] and fine-grained access control to a key-value store in [45].

Mobile Computations. In my doctoral thesis, I tried to devise abstractions for programming wide area networks, with the Seal calculus, a core model of mobile computations [47], [128]. The Seal calculus was among the first to explore the design space of mobile languages from both theoretical and practical angles [41].

Towards Rigor in Experimental Computer Science. I have always been interested in improving the state of empirical evaluation in our field. This is one of the motivation for developping benchmarks for domains as different as software transaction [99], real-time computing [27], [152], web applications [79] and concurrent programming [74]. More recently, I started advocating for rigor and repeatability of experimental computer science [76], [194]. Science advance faster when one can build on existing results, and when new ideas can easily be measured against the state of the art. This is exceedingly difficult in an environment that does not reward the production of reusable software artifacts. *Repeatability* can be summed up as a validation of the claims made in a paper by re-running a bundled software artifact prepared by the paper authors. Repeatability is a cheap and easy test which clarifies the scientific contribution of a paper. It should become a standard feature of the dissemination of research results. Together with Shriram Krishnamurthi, I have led the effort on including artifact evaluation as a standard part of major conferences [22].

Publications

Journals

[1] J. Belyakova, B. Chung, R. Tate, J. Vitek. Decidable Subtyping of Existential Types for Julia. In *Proc. ACM PL* (**PACMPL(PLDI)**), 2024.

[2] M. Kalpesh Mehta, S. Krynski, H. Musso Gualandi, M. Thakur, J. Vitek. Reusing Just-in-Time Compiled Code. In *Proc. ACM PL* (**PACMPL(OOSPLA)**), 2023. doi:10.1145/3622839

[3] A. Goel, P. Donat-Bouillud, F. Krikava, C;M. Kirsch, J. Vitek: What we eval in the shadows: a large-scale study of eval in R programs. In *Proc. of the ACM PL* (**PACMPL(OOPSLA)**), 2021. doi:10.1145/3485502

[4] A. Goel, J. Jecmen, S. Krynski, O. Flückiger, J. Vitek: Promises are made to be broken: migrating R to strict semantics. In *Proc. of the ACM PL* (**PACMPL(OOPSLA)**), 2021. doi:10.1145/3485478

[5] A. Pelenitsyn, J. Belyakova, B. Chung, R. Tate, J. Vitek: Type stability in Julia: avoiding performance pathologies in JIT compilation. In *Proc. of the ACM PL* (**PACMPL(OOPSLA)**) 2021. doi:10.1145/3485527

[6] A. Barriere, O. Flückiger, S. Blazy, D. Pichardie, J. Vitek. Formally Verified Speculation and Deoptimization in a JIT Compiler. In *Proc. of the ACM PL* (**PACMPL(POPL)**), 2021. doi:10.1145/3434327

[7] J. Belyakova, B. Chung, J. Gelinas, J. Nash, R. Tate, J. Vitek. World Age in Julia: Optimizing Method Dispatch in the Presence of Eval. In *Proc. of the ACM PL* (**PACMPL(OOPSLA)**), 2020.

[8] A. Turcotte, A. Goel, F. Krikava, J. Vitek. Designing Types for R, Empirically. In *Proc. of the ACM PL* (**PACMPL(OOPSLA)**), 2020.

[9] O. Flückiger, G. Chari, M. Yee, J. Jecmen, J. Hain, J. Vitek. Contextual Dispatch for Function Specialization In *Proc. of the ACM PL* (**PACMPL-(OOPSLA)**), 2020.

[10] F. Krikava, H. Muller, J. Vitek. Scala Implicits are Everywhere: A large-scale study of the use of Implicits in the wild. In *Proc. of the ACM PL* (**PACMPL(OOPSLA)**), 2019. doi:10.1145/3360589

[11] A. Goel, J. Vitek. On the Design, Implementation and Use of Laziness in R. In *Proc. of the ACM PL* (**PACMPL(OOPSLA)**), 2019. doi:10.1145/3360579

[12] E. Berger, C. Hollenbeck, P. Maj, O. Vitek, J. Vitek. On the Impact of Programming Languages on Code Quality. In *ACM Trans. on Prog. Lang.*, (**TOPLAS**), 2019. doi:10.1145/3340571

[13] Y.Ġokul, K. Dantu, S. Ko, L. Ziarek, J. Vitek. Can Android Run on Time? Extending and Measuring the Android Platform's Timeliness. In *ACM Trans. Embedded Comput. Syst.*, (**TECS**), 2019. doi:10.1145/3289257

[14] L. Andersen, V. St-Amour, J. Vitek, M. Felleisen. Feature-Specific Profiling. In *ACM Trans. on Prog. Lang.* (**TOPLAS**), 2019. doi:10.1145/3275519

[15] B. Greenman, A. Takikawa, M. New, D. Felty, R. Findler, J. Vitek, M. Felleisen. How to Evaluate the Performance of Gradual Type Systems. In *Journal of Functional Programming*, (**JFP**), 2019. doi:10.1017/S0956796818000217

[16] Y. Zakowski, D. Cachera, D. Demange, G. Petri, D. Pichardie, S. Jagannathan, J. Vitek. Verifying a Concurrent Garbage Collector with a Rely-Guarantee Methodology. In *Journal of Automated Reasoning*, (**JAR**), 2018. doi:10.1007/978-3-319-66107-0_31

[17] F. Zappa Nardelli, J. Belyakova, A. Pelenitsyn, B. Chung, J. Bezanson, J. Vitek. Julia subtyping: a rational reconstruction. In *Proc. ACM Prog. Lang.*, (**OOPSLA**), 2018. doi:10.1145/3276483

[18] J. Bezanson, J. Chen, B. Chung, S. Karpinski, V. Shah, J. Vitek, L. Zoubritzky./ Julia: dynamism and performance reconciled by design. In *Proc. ACM Prog. Lang.*, (**OOPSLA**), 2018. doi:10.1145/3276490

[19] O. Flückiger, G. Scherer, M. Yee, A. Goel, A. Ahmed, J. Vitek. Correctness of speculative optimizations with dynamic deoptimization. In *Proc. ACM Prog. Lang.*, (**POPL**), 2018. doi:10.1145/3158137

[20] S. Clebsch, J. Franco, S. Drossopoulou, A. Mingkun Yang, T. Wrigstad, J. Vitek. Orca: GC and type system co-design for actor languages. In *Proc. ACM Program. Lang.*, (**OOPSLA**), 2017. doi:10.1145/3133896

[21] C. Lopes, P. Maj, P. Martins, D. Yang, J. Zitny, H. Sajnani. J. Vitek. Déjà Vu: A Map of Code Duplicates on GitHub. In *Proc. ACM Program. Lang.*, (**OOPSLA**), 2017, doi:10.1145/3133908.

[22] S. Krishnamurthi, J. Vitek. The real software crisis: repeatability as a core value. In *Commun. ACM*, (**CACM**), 58(3), 2015. doi:10.1145/2658987

[23] S. Jagannathan, V. Laporte, G. Petri, D. Pichardie, J. Vitek. Atomicity Refinement for Verified Compilation. In *ACM Transaction on Programming Languages and Systems*, (**TOPLAS**), 2014. doi:10.1145/2601339

[24] F. Meawad, K. Iyer, M. Schoeberl and J. Vitek. Micro-transactions for concurrent data structures. In *Concurrency and Computation: Practice and Experience* (**CCPE**), 25(16): 2252-–2268, 2013. doi:10.1002/cpe.2985

[25] J. Dolby, C. Hammer, D. Marino, F. Tip, M. Vaziri, J. Vitek. A Data-Centric Approach to Synchronization. *ACM Transaction on Programming Languages and Systems*, (**TOPLAS**) 48 pages, 34(1):4, 2012. doi:10.1145/2160910.2160913

[26] T. Kalibera, F. Pizlo, T. Hosking, J. Vitek. Scheduling real-time garbage collection on uniprocessors. In *ACM Transaction on Computer Systems*, (**TOCS**), 29:3, pp. 8:1–8:29, 2011.

[27] T. Kalibera, J. Hagelberg, P. Maj, F. Pizlo, B. Titzer, and J. Vitek. A family of real-time Java benchmarks. In *Concurrency and Computation: Practice and Experience*, (**CS:PE**), 23(14), pp. 1679–1700, 2011. doi:10.1002/cpe.1677

[28] J. Spring, F. Pizlo, J. Privat, R. Guerraoui, J. Vitek. Reflexes: Abstractions for Integrating Highly Responsive Tasks into Java Applications. In *ACM Transactions in Embedded Computing Systems* (**TECS**), 2010. 28 pages.

[29] J. Baker, A. Cunei, T. Kalibera, F. Pizlo, J. Vitek. Accurate Garbage Collection in Uncooperative Environments. In *Concurrency and Computation: Practice and Experience*, (**CC:PE**), 21(12), pp. 1572–1606, 2009.

[30] T. Zhao, J. Baker, J. Hunt, J. Noble and J. Vitek. Implicit Ownership Types for Memory Management, In *Science of Computer Programming*, (**SCP**), 71, pp. 213–241, 2008.

[31] A. Cunei, J. Vitek. An Efficient ad Flexible Toolkit for Composing Customized Method Dispatchers. In *Software Practice and Experience*, (**SPE**), 38(1), pp. 33–73, 2008.

[32] C. Andrea, Y. Coady, C. Gibbs, J. Noble, T. Zhao, J. Vitek. Scoped Types and Aspects for Real-time Java Memory Management. In *Real-time Systems Journal*, (**RSJ**) pp. 1–44, October, 2007.

[33] A. Armbuster, J. Baker, A. Cunei, C. Flack, D. Holmes, F. Pizlo, E. Pla, M. Prochazka, J. Vitek. A Real-time Java Virtual Machine with Applications in Avionics. In *ACM Transactions in Embedded Computing Systems (TECS)*, (**TECS**) 7(1), pp. 1–49 pages, 2007.

[34] C. Grothoff, J. Palsberg, J. Vitek. Encapsulating Objects with Confined Types. In *ACM Transactions on Programming Languages and Systems*, 29(6), (**TOPLAS**), 41 pages, 2007.

[35] O. Vitek, B. Craig, C. Bailey-Kellog, J. Vitek. Inferential backbone assignment for sparse data. In *Journal of Biomolecular NMR*, (**JBNMR**), 35(3), pp. 187–208, Springer, 2006.

[36] B. Cărbunar, A. Grama, J. Vitek, O. Cărbunar. Redundancy and Coverage Detection in Sensor Networks. In *ACM Transaction on Sensor Networks*, (**TSN**), pp. 94–128, 2(1), 2006.

[37] T. Zhao, J. Palsberg, J. Vitek. Type-based Confinement. In *The Journal of Functional Programming*, (**JFP**), pp 83–128, 16(1), January 2006.

[38] O. Vitek, C. Bailey-Kellogg, B. Craig, P. Kuliniewicz, J. Vitek. Reconsidering Complete Search Algorithms for Protein Backbone NMR Assignment. In **Bioinformatics**, 21, pp. 230–236, September 2005.

[39] S. Jagannathan, J. Vitek, A. Welc, T. Hosking, A Transactional Object Calculus. In *Science of Computer Programming*, (**SCP**) Elsevier, pp. 164–186, 57(2), August 2005.

[40] K. Palacz, J. Baker, C. Flack, C. Grothoff, H. Yamauchi and J. Vitek. The Ovm customizable intermediate representation. In *Science of Computer Programming*, (**SCP**), pp. 357–378, 57(3) Elsevier, September 2005.

[41] G. Castagna, J. Vitek and F. Zappa Nardeli. The Seal calculus. In *Information and Computation*, (**I&C**) Elsevier, 201(1), pp. 1–54, August 2005.

[42] O. Vitek, J. Vitek, B. Craig and C. Bailey-Kellogg. Model-based assignment and inference of protein backbone nuclear magnetic resonances. In *Statistical Applications in Genetics and Molecular Biology*, (**SAGMB**), Berkeley Electronic Press, Volume 1, Issue 1, 2004.

[43] B. Carbunar, M. T. Valente and J. Vitek. Lime revisited. In *Mathematical Structures in Computer Science*, (**MSCS**) Cambridge University Press, 14(3), pp. 397–419, 2004.

[44] P. Sewell and J. Vitek. Secure composition of untrusted code: box-π, wrappers and causality types. In *Journal of Computer Security*, (**JCS**), IOS Press, 11, pp. 135–188, 2003.

[45] J. Vitek, C. Bryce and M. Oriol. Coordinating agents with secure spaces. In *Science of Computer Programming*, (**SCP**), Elsevier, 46, pp. 163–193, 2002.

[46] J. Vitek and B. Bokowski. Confined types for Java. In *Software Practice and Experience*, (**SPE**), Wiley, 31, pp. 507–532, 2001.

[47] C. Bryce and J. Vitek. The JavaSeal mobile agent kernel. In *Autonomous Agents and Multi-Agent Systems*, (**AAMAS**), Kluwer, 4, pp. 359–384, 2001.

[48] R. N. Horspool and J. Vitek. Static analysis of PostScript code. In *Journal of Computer Languages*, (**JCL**), Pergamon Press, 19, pp. 65-78, 1993.

[49] G. Kappel, J. Vitek, O. Nierstrasz, B. Junod and M. Stadelmann. Scripting applications in the public administration domain. In **SIGOIS Bulletin**, 10, pp. 21-32, 1992.

Refereed Conference Proceedings

[50] P. Maj, S. Muroya, K. Siek, L. Di Grazia, J. Vitek. The Fault in Our Stars: Designing Reproducible Large-scale Code Analysis Experiments. In *European Conference on Object Oriented Programming* (**ECOOP**), 2024.

[51] A. Turcotte, P. Donat-Bouillud, F. Krikava, J. Vitek. signatr: A Data-Driven Fuzzing Tool for R. In *International Conference on Software Language Engineering* (**SLE**), 2022. doi:10.1145/3567512.3567530

[52] O. Flückiger, J. Jecmen, S. Krynski, J. Vitek. Deoptless: Speculation with Dispatched On-Stack Replacement and Specialized Continuations. In *Programming Language Design and Implementation Conference* (**PLDI**), 2022. doi:10.1145/3519939.3523729

[53] A. Goel, J. Vitek. First-Class Environments in R. In *Symposium on Dynamic Languages (DLS)*, 2021. doi:10.1145/3486602.3486768

[54] O. Flückiger, S. Krynski, A. Wälchli, J. Vitek. Sampling Optimized Code for Type Feedback. In *Dynamic Language Symposium* (**DLS**), 2020.

[55] O. Fluckiger, M. Yee, G. Chari, J. Hain, J. Jecmen, J. Vitek. R Melts Brains: An IR for First-Class Environments and Lazy Effectful Arguments. In *Dynamic Language Symposium* (**DLS**), 2019. doi:10.1145/3276945.3276946

[56] B. Chung, F. Zappa Nardelli, J. Vitek. Julia's efficient algorithm for subtyping unions and covariant tuples. In *European Conference on Object Oriented Programming* (**ECOOP**), 2019. doi: 10.4230/LIPIcs.ECOOP.2019.2.

[57] G. Chari, J. Pimas, O. Flückiger, J. Vitek. Self-Contained Development Environments. In *Dynamic Language Symposium*, (**DLS**), 2018. doi:10.1145/3276945.3276948

[58] B. Chung, P. Li, F. Zappa Nardelli, J. Vitek. KafKa: Gradual Typing for Objects. In *European Conference on Object Oriented Programming* (**ECOOP**), Amsterdam, July 2018. doi:10.4230/LIPIcs.ECOOP.2018.12.

[59] J. Franco, S. Clebsch, S. Drossopoulou, J. Vitek, T. Wrigstad. Correctness of a Concurrent Object Collector for Actor Languages. In *European Symposium on Programming*, (**ESOP**), 2018. doi:10.1007/978-3-319-89884-1_31.

[60] F. Krikava, J. Vitek. Tests from traces: automated unit test extraction for R. In *International Symposium on Software Testing and Analysis*, (**ISSTA**), Amsterdam, 2018. doi:10.1145/3213846.3213863

[61] T. Anderson, H. Liu, L. Kuper, E. Totoni, J. Vitek, T. Shpeisman. Parallelizing Julia with a Non-Invasive DSL. In *European Conference on Object Oriented Programming* (**ECOOP**), 2017. doi:10.4230/LIPIcs.ECOOP.2017.4

[62] Y. Zakowski, D. Cachera, D. Demange, G. Petri, D. Pichardie, S. Jagannathan, J. Vitek. Verifying a Concurrent Garbage Collector Using a Rely-Guarantee Methodology. In *Interactive Theorem Proving*, (**ITP**), 2017. doi:10.1007/978-3-319-66107-0_31

[63] Y. Yan, K. Dantu, S. Ko, J. Vitek, L. Ziarek. Making Android Run on Time. In *Real-Time and Embedded Technology and Applications Symposium* (**RTAS**), 2017. doi:10.1109/RTAS.2017.38

[64] A. Takikawa, D. Feltey, B. Greenman, M. New, J. Vitek, M. Felleisen. Is sound gradual typing dead? In *ACM Symposium on Principles of Programming Languages* (**POPL**), 2016. doi:10.1145/2837614.2837630

[65] G. Petri, J. Vitek, S. Jagannathan, Cooking the Books: Formalizing JMM Implementation Recipes. In *European Conference on Object Oriented Programming* (**ECOOP**), Prague, July 2015.

[66] G. Richards, F. Zappa Nardelli, J. Vitek. Concrete Types for TypeScript. In *European Conference on Object Oriented Programming* (**ECOOP**), Prague, July 2015.

[67] T. Kalibera, P. Maj, F. Morandat, J. Vitek. A Fast Abstract Syntax Tree Interpreter for R. In *Conference on Virtual Execution Environments* (**VEE**), Salt Lake City, March 2014.

[68] G. Richards, C. Hammer, S. Jagannathan, F. Zappa Nardelli and J. Vitek. Flexible Access Control Policies with Delimited Histories and Revocation In *Conference on Object-Oriented Programming Systems, Languages and Applications* (**OOPSLA**), Indianapolis, October 2013.

[69] D. Marino, C. Hammer, J. Dolby, M. Vaziri, F. Tip, and J. Vitek, Detecting deadlock in programs with data-centric synchronization. In *International Conference on Software Engineering* (**ICSE**), pp. 322-331, San Francisco, May 2013.

[70] Z. DeVito, J. Hegarty, A. Aiken, P. Hanrahan, and J. Vitek. Terra: a multi-stage language for high-performance computing. In *Programming Language Design and Implementation Conference* (**PLDI**) pp. 105-116, Seattle, June 2013.

[71] D. Demange, V. Laporte, L. Zhao, S. Jagannathan, D. Pichardie, and J. Vitek. Plan B: a buffered memory model for Java. In *ACM Symposium on Principles of Programming Languages* (**POPL**), pp. 329-342, Rome, January 2013.

[72] A. Bouakaz, J.-P. Talpin and J. Vitek. Affine Data-Flow Graphs for the Synthesis of Hard Real-Time Applications. In *Conference on Application of Concurrency to System Design* (**ACSD**) pp 183-192, Hamburg, July 2012.

[73] F. Morandat, B. Hill, L. Osvald and J. Vitek, Evaluating the Design of the R Language. In

European Conference on Object-Oriented Programming (**ECOOP**) Beijing, June 2012.

[74] T. Kalibera, M. Mole, R. E. Jones and J. Vitek. A black-box approach to understanding concurrency in DaCapo. In *Conference on Object-Oriented Programming Systems, Languages and Applications* (**OOPSLA**), pp. 335-354, Tucson, October 2012.

[75] F. Meawad, G. Richards, F. Morandat and J. Vitek. Eval begone!: semi-automated removal of eval from JavaScript programs. In *Conference on Object-Oriented Programming Systems, Languages and Applications* (**OOPSLA**), pp. 607-620, Tucson, October 2012.

[76] J. Vitek, T. Kalibera. Repeatability, reproducibility, and rigor in systems research. In *International Conference on Embedded Software* (**EMSOFT**), Taipei, October, 2011.

[77] J. Vitek. Virtualizing real-time embedded systems with Java. In *Design Automation Conference* (**DAC**), San Diego, June, 2011.

[78] A. Milanova, J. Vitek. Static Dominance Inference. In *Conference on Objects, Models, Components, Patterns* (**TOOLS**), Zurich, June, 2011.

[79] G. Richards, A. Gal, B. Eich, J. Vitek. Automated Construction of JavaScript Benchmarks. In *Conference on Object-Oriented Programming Systems, Languages and Applications* (**OOPSLA**), pages 677–694, 2011.

[80] G. Richards, C. Hammer, B. Burg, J. Vitek. The Eval that Men Do: A Large-scale Study of the Use of Eval in JavaScript Applications. In *European Conference on Object-Oriented Programming* (**ECOOP**), pages 52–78, Lancaster July 2011.

[81] F. Pizlo, E. Blanton, P. Maj, J. Vitek, L. Ziareck. Schism: Fragmentation-Tolerant Real-Time Garbage Collection. In *Programming Language Design and Implementation Conference* (**PLDI**), Toronto, June 2010.

[82] G. Richards, S. Lesbrene, B. Burg, J. Vitek. An Analysis of the Dynamic Behavior of JavaScript Programs. In *Programming Language Design and Implementation Conference* (**PLDI**), Toronto, June 2010.

[83] M. Vaziri, F. Tip, J. Dolby, C. Hammer, J. Vitek. A Type System for Data-Centric Synchronization. In In *European Conference on Object-Oriented Programming* (**ECOOP**), pages 304–328, Maribor, Slovenia, June, 2010.

[84] F. Pizlo, L. Ziareck, E. Blanton, P. Maj, J. Vitek. High-level Programming of Embedded Hard Real-Time Devices. In *European Conference on Computer Systems* (**EUROSYS**), Paris, April 2010.

[85] T. Wrigstad F. Zappa Nardelli, S. Lebresne, J. Ostlund, J. Vitek. Integrating of Typed and Untyped Code in a Scripting Language. In *ACM Symposium on Principles of Programming Languages* (**POPL**), Madrid, January 2010.

[86] M. Schoeberl, F. Brandner, J. Vitek. RTTM: Real-Time Transactional Memory. In *Symposium on Applied Computing, Real-Time Systems Track* (**SAC**), Sierre, March 2010.

[87] T. Kalibera, F. Pizlo, A. Hosking, J. Vitek. Scheduling Hard Real-time Garbage Collection. In *Real-Time Systems Symposium* (**RTSS**), Washington D.C., December 2009.

[88] B. Bloom, J. Field, N. Nystrom, J. Ostlund, G. Richards, R. Strnisa, J. Vitekand T. Wrigstad. Thorn—Robust, Concurrent, Extensible Scripting on the JVM. In *Conference on Object-Oriented Programming Systems, Languages and Applications* (**OOPSLA**), 20 p., Orlando, October 2009.

[89] T. Kalibera, M. Prochazka, F. Pizlo, J. Vitek, M. Zulianello, M. Decky. Real-time Java in Space: Potential Benefits and Open Challenges. In *DAta Systems In Aerospace* (**DASIA**), 10 pp., Istanbul, June 2009.

[90] T. Wrigstad, F. Pizlo, F. Meawad, L. Zhao, J. Vitek. Loci: Simple Thread-Locality for Java. In *European Conference on Object Oriented Programming* (**ECOOP**), Genova, June 2009.

[91] A. Cunei, R. Guerraoui, J. Spring, J. Privat, J. Vitek. High-Performance Transactional Event Processing. In *the International Conference on Coordination Models and Languages* (**COORDINATION**), pp. 27–46, Madrid, June 2009.

[92] J. Auerbach, J. H. Spring, D. Bacon, R. Guerraoui, J. Vitek. A Unified Restricted Thread Programming Model for Java. In *Conference on Languages, Compilers, and Tools for Embedded Systems* (**LCTES**), Tucson, pp. 1–11, June 2008.

[93] F. Pizlo, J. Vitek. Memory Management for Real-time Java: State of the Art. In *International Symposium on Object-oriented Real-Time Distributed Computing* (**ISORC**), pp. 248–254, Orlando, May 2008.

[94] M. Hirtzel, B. Bloom, N. Nystrom, J. Vitek. Matchete: Paths through the Pattern Matching Jungle. In *Symposium on Practical Aspects of Declarative Languages* (**PADL**), San Francisco, pp. 150–166, January 2008.

[95] J. H. Spring, J. Privat, R. Guerraoui, J. Vitek. StreamFlex – High performance stream programming in Java. *Conference on Object-Oriented Programming Systems, Languages and Applications* (**OOPSLA**), pp. 211–228, Montreal, October 2007.

[96] F. Pizlo, A. Hosking, J. Vitek. Hierarchical Real-time Garbage Collection. In *Conference on Languages, Compilers, and Tools for Embedded Systems* (**LCTES**), pp. 123–133, San Diego, June 2007.

[97] J.H. Spring, F. Pizlo, R. Guerraoui, J. Vitek. Reflexes: Abstractions for Highly Responsive Systems. In *Conference on Virtual Execution Environments* (**VEE**), pp. 191–201, San Diego, June 2007.

[98] J. Baker, A. Cunei, F. Pizlo, J. Vitek. Accurate Garbage Collection in Uncooperative Environments with Lazy Pointer Stacks. In *Conference on Compiler Construction* (**CC**), pp. 64–79, Braga,q March 2007.

[99] M. Kalpka, R. Guerraoui, J. Vitek. STMBench7: A Benchmark for Software Transactional Memory. In *European Conference on Computer Systems* (**EUROSYS**), Lisbon, pp. 315–324, March 2007.

[100] F. Pizlo, J. Vitek. An Empirical Evalutation of Memory Management Alternatives for Real-time Java. In *27th Real-Time Systems Symposium* (**RTSS**), pp. 35–46, Rio de Janeiro, December 2006.

[101] H. Yamauchi, J. Vitek. Combining Offline and Online Optimizations: Register Allocation and Method Inlining. In *ASIAN Symposium on Programming Languages and Systems* (**APLAS**), pp. 307–322, Sydney, November 2006.

[102] C. Andrea, Y. Coady, C. Gibbs, J. Noble, J. Vitek, T. Zhao. Scoped Types and Aspects for Real-Time Systems. In *European Conference on Object Oriented Programming* (**ECOOP**), pp. 124–147, Nantes, July 2006.

[103] A. Cunei, J. Vitek. A New Approach to Real-time Checkpointing. In *Conference on Virtual Execution Environments* (**VEE**), pp. 68–77, Ottawa, June 2006.

[104] J. Baker, A. Cunei, C. Flack, F. Pizlo, M. Prochazka, J. Vitek, A. Armbuster, E. Pla, D. Holmes. Real-time Java in Avionics Applications. In *Real-Time and Embedded Technology and Applications Symposium* (**RTAS**), pp. 384–396, San Jose, April 2006.

[105] J. Manson, J. Baker, A. Cunei, S. Jagannathan, M. Prochazka, B. Xin, J. Vitek. Preemptible Atomic Regions for Real-time Java. In *Real-Time Systems Symposium* (**RTSS**), pp. 62–71, Miami, December 2005.

[106] A. Cunei, J Vitek, PolyD: A Flexible Dispatching Framework, In *Conference on Object-Oriented Programming Systems, Languages and Applications* (**OOPSLA**), pp. 487–504, San Diego, October 2005.

[107] O. Vitek, C. Bailey-Kellogg, B. Craig, P. Kuliniewicz, J. Vitek, Reconsidering Complete Search Algorithms for Protein Backbone NMR Assignment. In *European Conference on Computational Biology* (**ECCB**), pp. 236–245, Vienna, September 2005.

[108] R. Gopalakrishna, E. Spafford, J. Vitek. Efficient Intrusion Detection using Automaton Inlining. In *Symposium on Security and Privacy* (**S&P**), pp. 18–31, Oakland, May 2005.

[109] T. Zhao, J. Noble, J. Vitek, Scoped Types for Real-time Java, In *Real-Time Systems Symposium* (**RTSS**), pp. 241–251, Lisbon, December 2004.

[110] B. Carbunar, I. Ioannidis, A. Grama, J. Vitek. A Secure Crediting Protocol for Hybrid Cellular and Ad-Hoc Networks. In *Conference on E-Business and Telecommunication Networks* (**ICETE**). pp. 142–149, Setubal, August 2004.

[111] B. Carbunar, A. Grama, J. Vitek. Coverage Preserving Redundancy Elimination in Sensor Networks. In *Conference on Sensor and Ad-Hoc Communications and Networks* (**SECON**). 2004.

[112] B. Carbunar, A. Grama and J. Vitek. Distributed and Dynamic Voronoi Overlays for Coverage Detection and Distributed Hash Tables in Ad-Hoc Networks. In *International Conference on Parallel and Distributed Systems* (**ICPADS**), pp. 549–559, Newport Beach, July 2004.

[113] F. Pizlo, J. Fox, D. Holmes and J. Vitek. Real-Time Java scoped memory: design patterns and semantics. In *International Symposium on Object-oriented Real-Time Distributed Computing* (**ISORC**), pp. 101–112, Vienna, May 2004.

[114] J. Vitek, S. Jagannathan, A. Welc and A.L. Hosking. A semantic framework for designer transactions. In *European Symposium on Programming* (**ESOP**), pp. 249–263, Barcelona, April 2004.

[115] S. Jagannathan and J. Vitek. Optimistic concurrency semantics for transactions in coordination languages. In *Conference on Coordination Models and Languages* (**COORDINATION**), pp. 183–198, Pisa, March 2004.

[116] T. Chothia, D. Dugganand J. Vitek, Principals, Policies and Keys in a Secure Distributed Programming Language. In *Computer Security Foundations* (**CSF**), pp. 170–180 Turku, July, 2003.

[117] T. Zhao, J. Palsberg and J. Vitek. Lightweight confinement for featherweight Java. In *Conference on Object-Oriented Programming Systems and Languages* (**OOPSLA**), pp. 135–148, San Diego, October 2003.

[118] T. Chothia, D. Duggan and J. Vitek. Type-based distributed access control. In *Computer Security Foundations Workshop* (**CSFW**), pp. 170 – 187, Pacific Grove, July 2003.

[119] K. Palacz and J. Vitek. Subtype tests in real time. In *European Conference on Object Oriented Programming* (**ECOOP**), pp. 378–404, Darmstadt, July 2003.

[120] K. Palacz, J. Baker, C. Flack, C. Grothoff, H. Yamauchi and J. Vitek. Engineering a customizable intermediate representation. In *Workshop on Interpreters, Virtual Machines and Emulators* (**IVME**), pp. 1–12, San Diego, June 2003.

[121] K. Palacz, G. Czaikowski, L. Daynes and J. Vitek. Incommunicado: a communication substrate for Isolates. In *Conference on Object-Oriented Programming Systems and Languages* (**OOPSLA**), pp. 262–274, Seattle, November 2002.

[122] B. Carbunar, M. T. Valente and J. Vitek. Lime revisited. In *International Conference on Mobile Agents* (**MA**), pp. 54–69, Atlanta, December 2001.

[123] C. Grothoff, J. Palsberg and J. Vitek. Encapsulating objects with confined types. In *Conference on Object-Oriented Programming Systems and Languages* (**OOPSLA**), pp. 241–255, Florida, October 2001.

[124] P. Sewell and J. Vitek. Secure composition of untrusted code: wrappers and causality types. In *Computer Security Foundations Workshop* (**CSFW**), pp. 269–284, Cambridge, July 2000.

[125] P. Sewell and J. Vitek. Secure composition of insecure components. In *Computer Security Foundations Workshop* (**CSFW**), pp. 136–150, Mordano, June 1999.

[126] C. Bryce, M. Oriol and J. Vitek. Secure object spaces: a coordination model for agents. In *International Conference on Coordination Models and Languages* (**COORDINATION**), pp. 4–20, Amsterdam, April 1999.

[127] B. Bokowski and J. Vitek. Confined types. In *ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Applications* (**OOPSLA**), pp. 82–97, Denver, October 1999.

[128] J. Vitek and C. Bryce. Security for mobile code: the JavaSeal experiment. In *Agent Systems and Applications Mobile Agents* (**ASA/MA**), pp. 103–118, Palm Springs, October 1999.

[129] Q. Bradley, R. N. Horspool and J. Vitek. JAZZ: An efficient compressed format for Java archive files. In *IBM CASCON Conference* (**CASCON**), pp. 294–302, Toronto, December 1998.

[130] J. Noble, J. Vitek and J. Potter. Flexible alias protection. In *European Conference on Object-Oriented Programming* (**ECOOP**), pp. 158–185, Brussels, July 1998.

[131] J. Vitek, R.N. Horspool and A. Krall. Efficient type inclusion tests. In *Conference on Object-Oriented Programming Systems, Languages and Applications* (**OOPSLA**), pp. 142–157, San Jose, October 1997.

[132] A. Krall, J. Vitek and R.N. Horspool. Near optimal hierarchical encoding of types. In *In European Conference on Object-Oriented Programming* (**ECOOP**), pp. 128–146, Jyvaskyla, June 1997.

[133] A. Krall and J. Vitek. On extending Java. In *Joint Modular Languages Conference (***JMLC***)*, pp. 321–325, Linz, March 1997.

[134] J. Vitek and R. N. Horspool. Compact dispatch tables for dynamically typed object oriented languages. In *Conference on Compiler Construction* (**CC**), pp. 309–326, Linkoping, April 1996.

[135] K. Driesen, U. Hölzle and J. Vitek. Message dispatch on pipelined processors. In *European Conference on Object-Oriented Programming* (**ECOOP**), pp. 253–283, Åarhus, August 1995.

[136] J. Vitek and R. N. Horspool. Taming message passing: efficient method look-up for dynamically typed languages. In *European Conference on Object-Oriented Programming* (**ECOOP**), pp. 432–449, Bologna, July 1994.

[137] J. Vitek, R.N. Horspool and J. Uhl. Compile-time analysis of object-oriented programs. In *Conference on Compiler Construction* (**CC**), pp. 236–250, Paderborn, October 1992.

[138] R. N. Horspool and J. Vitek. Static analysis of PostScript code. In *International Conference on Computer Languages* (**ICCL**), pp. 14–23, Oakland, April 1992.

Refereed Workshop Publications

[139] A. Goel, J. Vitek. RDT: A Dynamic Tracing Framework for R. In *Workshop on R Implementation, Optimization and Tooling* (**RIOT**), Toulouse, 2019.

[140] A. Turcotte, J. Vitek. In *Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems* (**ICOOOLPS**), London, 2019.

[141] B. Chung, J. Vitek. Monotonic Gradual Typing in a Common Calculus. In *Workshop on Formal Techniques for Java-like Programs*, (**FTfJP**), Amsterdam, 2018.

[142] J. Vitek. What can R learn from Julia. In **userR!**, Stanford, 2016.

[143] B. Chung, P. Li, J. Vitek. Static Typing Without Static Types - Typing Inheritance from the Bottom Up. In *Workshop on New Object-Oriented Languages*, (**NOOL**), Amsterdam, 2016.

[144] A. Takikawa, D. Feltey, B. Greenman, M. New, J. Vitek, M. Felleisen. Position Paper: Performance Evaluation for Gradual Typing. In *Scripts to Programs* (**STOP**), Prague, July 2015.

[145] R. Macnak, F. Morandat, B. Hill, L. Osvald and J. Vitek. TraceR: A framework for understanding R performance. In *International R Users Meeting* (**UseR!**), Nashville, June, 2012.

[146] F. Meawad, K. Iyer, M. Schoeberl, J. Vitek. Real-Time Wait-free Queues using Micro-Transactions. In *International Workshop on Java Technologies for Real-time and Embedded Systems* (**JTRES**), York, September 2011.

[147] F. Meawad, B. Macnak, J. Vitek. Collecting Transactional Garbage. In TRANSACT, June, 2011.

[148] N. Kidd, S. Jagannathan, J. Vitek. One Stack to Run Them All: Reducing Concurrent Analysis to Sequential Analysis Under Priority Scheduling. In *SPIN Workshop on Model Checking of Software* (**SPIN**), Enschede, September 2010.

[149] D. Tang, A. Plsek, J. Vitek. Static Checking of Safety Critical Java Annotations. In *Workshop on Java Technologies for Real-time and Embedded Systems* (**JTRES**), Prague, September 2010.

[150] A. Plsek, L. Zhao, V. Sahin, D. Tang, T. Kalibera, J. Vitek. Developing Safety Critical Java applications with oSCJ/L0. In *Workshop on Java Technologies for Real-time and Embedded Systems* (**JTRES**), Prague, September 2010.

[151] T. Kalibera, P. Parizek, G. Haddad, G. Leavens, J. Vitek. Challenge Benchmarks for Verification of Real-time Programs. In *Workshop on Programming Languages meets Program Verification* (**PLPV**), 6 pages, Madrid, January 2010.

[152] T. Kalibera, J. Hagelberg, F. Pizlo, A. Plsek, B. Titzer, J. Vitek. CDx: A Family of Real-time Java Benchmarks. In *Workshop on Java Technologies for Real-time and Embedded Systems* (**JTRES**), Madrid, September 2009.

[153] L. Zhao, D. Tang, J. Vitek. A Technology Compatibility Kit for Safety Critical Java In *Workshop on Java Technologies for Real-time and Embedded Systems* (**JTRES**), Madrid, September 2009.

[154] F. Pizlo, L. Ziarek, J. Vitek. Towards Java on Bare Metal with the Fiji VM. In *Workshop on Java Technologies for Real-time and Embedded Systems* (**JTRES**), Madrid, September 2009.

[155] T. Wrigstad, P. Eugster, J. Field, N. Nystrom, J. Vitek. Software Hardening: A Research Agenda. In *Workshop on Script to Program Evolution* (**STOP**), Genoa, July 2009.

[156] S. Lebresne, G. Richards, J. Östlund, T. Wrigstad, J. Vitek. Understanding the Dynamics of JavaScript. In *Workshop on Script to Program Evolution* (**STOP**), Genoa, July 2009.

[157] J. Hunt, D. Locke, K. Nilsen, M. Schoeberl, J. Vitek. Java for Safety-Critical Applications. In *Certification of Safety-Critical Software Controlled Systems* (**SafeCert**), York, March 2009.

[158] M. Schoeberl, J. Vitek. Garbage Collection for Safety Critical Java. In *Workshop on Java Technologies for Real-time and Embedded Systems* (**JTRES**), Vienna, September 2007.

[159] I. Dragos, A. Cunei, J. Vitek. Continuation in the Java Virtual Machine. In *Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems* (**ICOOOLPS**), Berlin, July 2007.

[160] Y. Coady, C. Gibbs, M. Haupt, J. Vitek, H. Yamauchi. Towards a domain specific language for virtual machines. In *Domain-Specific Aspect Languages Workshop* (**DSAL**), Portland October 2006.

[161] J. Manson, S. Jagannathan, and J. Vitek. Dynamic Aspects for Runtime Fault Determination and Recovery. In *Dynamic Aspects Workshop* (**DAW**), Chicago, March 2005.

[162] F. Pizlo, M. Prochazka, S. Jaggannathan and J. Vitek. Transactional lock-free data structure for Real Time Java. In *Workshop on Concurrency and Synchronization in Java Programs*, St John's, Newfoundland, July 2004.

[163] B. Carbunar, M.T. Valente and J. Vitek. CoreLime: a coordination model for mobile agents. In *Workshop on Concurrency and Coordination*, Lipary, July 2001.

[164] J. Vitek and G. Castagna. Mobile computations and hostile hosts. In *Journées Francophones des Langages Applicatifs* (**JFLA**), pp. 113–132, Avoriaz, February 1999.

[165] J. Vitek. New Paradigms in distributed computing. In *European Research Seminar in Advanced Distributed Systems* (**ERSADS**), pp. 117–122, Zinal March 1997.

[166] J. Vitek. Secure object spaces. In *Workshop on Mobile Object Systems* (**MOS**), pp. 41-48, Linz, July 1996.

Book Chapters

[192] J. Vitek. The Case for the Three R's of Systems Research: Repeatability, Reproducibility and Rigor (Keynote). In *Conference on Virtual Execution Environments*, (**VEE**), Salt Lake City, March 2014.

[193] R. Hirschfeld, S. Krishnamurthi, J. Vitek. Foundations for Scripting Languages, pp 1–18, Dagstuhl Reports, 2192-5283, 2012.

[194] J. Vitek and T. Kalibera. R3: Repeatability, reproducibility and rigor. In ACM SIGPLAN Notices, 47(4a), pp. 30-36, April 2012.

[195] J. Vitek. Introduction to: The Myths of Object-Orientation. *European Conference on Object Oriented Programming* (**ECOOP**), July 2009.

[196] J. Vitek, C. Bryce and W. Binder. Designing JavaSeal, or how to make Java safe for agents. In *Electronic Commerce Objects*, pp. 105-126, U. of Geneva, 1998.

[197] J. Vitek. Compact dispatch tables for dynamically typed programming languages. In *Object Applications*, pp. 81-138, U. of Geneva, 1996.

[198] D. Konstantas, J.H. Morin and J. Vitek. MEDIA: A platform for the commercialization of electronic documents. In *Object Applications*, pp. 7-18, U. of Geneva, 1996.

[199] O. Nierstrasz, L. Dami, V. de Mey, M. Stadelmann, D. Tsichritzis and J. Vitek. Visual scripting – towards interactive construction of object-oriented applications. In *Object Management*, pp. 315-331, U. of Geneva, 1990.

[200] M. Stadelmann, G. Kappel and J. Vitek. VST: a scripting tool based on the UNIX shell. In *Object Management*, pp. 333-344, U. of Geneva, 1990.

[201] J. Vitek, B. Junod, O. Nierstrasz, S. Renfer and C. Werner. Events and sensors: enhancing the reusability of objects. In *Object Management*, pp. 345-356, U. of Geneva, 1990.

[202] G. Kappel, J. Vitek, O. Nierstrasz, S. Gibbs, B. Junod, M. Stadelmann and D. Tsichritzis. An object-based visual scripting environment. In *Object Oriented Development*, pp. 123-142, U. of Geneva, 1989.

Invited Lectures

* Prof. Strangelove. Or: How I learned to stop worrying and love dynamic languages. Invited talk at the DLS Conference, Cascais, 2023.
* On the design and foundations of dynamic languages for scientific computing. Invited talk at JuliCon, online 2021.
* On the design and foundations of dynamic languages for scientific computing. Keynote at the JuliCon Conference, online 2021.
* Fitzcarraldo as a Metaphor for Research. Keynote talk at the SPLASH 2020 Conference, online, 2020.
* R Melts Brains, or: How I Learned to Love Failing at Compiling R. Keynote talk at Why R? 2020 Conference, online, 2020.
- Getting everything wrong without doing anything right! (On the perils of large-scale analysis of Github data). Invited talk at the Curry On Conference, London, 2019.
* Adversarial Compilation. Keynote talk at Managed Programming Languages and Runtimes (MPLR), Athens, 2019.
* Meta-programming in Data Science. Keynote talk at the META Workshop, Athens, 2019.
* Reasoning about programs: Soundness revisited. Invited talk at the Prague computer science seminar, Prague, 2019.
- On the perils of large-scale analysis of Github data. Invited talk at PaperWeLove, London, 2019.
* The Beauty and the Beast – from Fortress to Julia. Keynote talk at the International Conference on Managed Languages and Runtimes (**ManLang**), Linz, 2018.
* Engineering your software engineering resarch career. Keynote talk at the *ICSE Doctoral Symposium*, Gothenburg, 2018.
* Data analysis for the masses. Keynote talk at the Federated Conference on Computer Science and Information Systems, Prague, 2017.
* This is not a Type: Gradual typing in practice. Keynote talk at the *Scala Symposium*, Amsterdam, 2016.
* Benchmarks killed the beast: Understanding JS performance for fun and profit. Keynote talk at the International Large Scale JavaScript Conference (**MLOC.JS**), Budapest, Hungary, 2015.
* Repeatability, reproducibility and rigor in CS research. Invited talk at the SIGPLAN Programming Language Mentoring Workshop, Mumbai, India, 2015.
* The Case for the Three R's of Systems Research: Repeatability, Reproducibility and Rigor. Keynote talk at the *Conference on Virtual Execution Environments*, Salt Lake City, March 2014.
* JavaScript Programmers Hate You: An ode to dynamic languages. Invited talk at the Workshop on Software Correctness and Reliability, Zurich, October 2013.
* Planet Dynamic or: How I Learned to Stop Worrying and Love Reflection. Invited talk at the *SIGPLAN Programming Languages Mentoring Workshop*, Rome, 2013.
* JavaScript Programmers Hate You. Keynote talk at at *Formal Techniques for Java-like Programs*, Montpellier, 2013.
* Planet Dynamic or: How I Learned to Stop Worrying and Love Reflection. Keynote talk at the *10th Asian Symposium on Programming Languages and Systems*, Kyoto, 2012.
* Repeatability, Reproducibility and Rigor. Invited talk at the *Conference on Languages Compilers and Tools for Embedded Systems*, Beijing, 2012.
* Thorn: Objects, Scripts and more... Invited talk at the *Concurrent Objects and Beyond* Symposium in Honor of Professor Akinori Yonezawa's 65th Birthday, Kobe, 2012.
* The Rise of Dynamic Language for Scientific Computing, Invited talk at the *Microsoft Faculty Summit*, Redmond, 2011.
* The Rise of Dynamic Language, Lecture at the *ECOOP Summer School*, Lancaster, 2011.

* Of Scripts and Programs: Tall tales, Urban Legends, and Future Prospects. Keynote talk at the *Analysis and Programming Languages for Web Applications and Cloud Applications*, Toronto, 2010.
* Is Java Ready for Real-time?, Invited talk at the *Midwest Verification Day* (MVD), U of Iowa, September, 10.
* Of Scripts and Programs: Tall tales, Urban Legends and Future Prospects, Keynote talk at the *Dynamic Languages Symposium*, Orlando, 2009.
* Programming Models for Concurrency and Real-time. Keynote talk at the *47th International Conference on Objects, Models, Components, Patterns (TOOLS)*, Zurich, 2009.
* Memory Management for Hard Real-time Systems. Invited talk at the *Workshop on Virtual Machines and Intermediate Languages for emerging modularization mechanisms*, Nashville, 2008.
* Programming models for Concurrency and Real-time. Invited talk at *XII Brazilian Symposium on Programming Languages*, Fortaleza, 2008.
* Programming models for Concurrency and Real-time. Invited talk at *Programming Language Approaches to Concurrency and Communication-cEntric Software*, Oslo, 2008.
* Semantics-based Intrusion Detection, Invited Talk at the *Foundations of Computer Security*, Chicago, 2005.
* Java for Hard Real-Time, Invited Talk at the *Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems*, Nantes, 2006.
* Advances in Intrusion Detection, Keynote talk at the *Program Analysis for Security and Safety Workshop (PASSWORD)*, Nantes, 2006.

Talks at International Meetings

* On the Impact of Programming Languages on Code Quality. OOPSLA, Athens, 2019.
* Do programming languages matter for correctness of code?, ETAPS Mentoring Workshop, Prague, 2019.
* Integrated Trustworthy Scripting Languages. *ONR TCP*, Seattle, 2018.
* The Beauty and the Beast — from Fortress to Julia. *IFIP Working Group on Language Design*, Antwerp, 2018.
* What You Need to Know about Performance Evaluation. *ECOOP Summer School*, Barcelona, 2017.
* Removing Abstraction Layers Dynamically. *Workshop on Forming an Ecosystem Around Software Transformation*, (**FEAST**), Vienna, 2016.
* 25 years of OO. *ECOOP Summer School*, Rome, 2016.
* Making R Run Fast. *Boston R Meet Up*, Boston, 2016.
* A fast abstract syntax tree interpreter for R. *Virtual Exectuion Envinronments*, Salt Lake City, UT, March, 2014.
* Planet Dynamic or: How I Learned to Stop Worrying and Love Reflection, NSA HCSS Conference, May 13
* R in Java. *UseR!*, Albacete, July 13.
* Understanding R. *Foundations of Scripting Languages*, Dagstuhl, January, 12.
* Repeatability, Reproducibility and Rigor. *IFIP Working Group 2.4* , Vadvestena, Sweden, 12.
* Taming the Tiger: How to scale R to bigger data. *Purdue Symposium on Statistics*, West Lafayette, June 12.
* Evaluating the Design of the R Language. *European Conference on Object Oriented Programming*, Beijing, June 12.
* CDx: A Family of Real-time Java Benchmarks. *Workshop on Java Technologies for Real-time and Embedded Systems* , Madrid, September 09.
* A Technology Compatibility Kit for Safety Critical Java. *Workshop on Java Technologies for Real-time and Embedded Systems*, Madrid, September 09.
* Software Hardening: A Research Agenda. *Workshop on Script to Program Evolution*, Genoa, July 09.
* Programming Real-time Embedded Systems in Java. Summer school part of the *Wroclaw Information Technology Initiative*, Wroclaw, May 09.
* Java for Safety-Critical Applications, *Certification of Safety-Critical Software Controlled Systems*, York, March 09.
* Large-Scale Embedded Programming, *Software Quality Symposium*, ETHZ, Zurich, 07.
* Programming Highly Responsive Systems, *IFIP Working Group 2.4*, Lake Arrowhead, CA, 07.
* Transactions and Composability: Transactions Considered Harmful? *IBM Workshop on Transactional Memory and Programming Technologies*, Armonk, March 07.
* Data-centric Synchronization, *IBM Workshop on Transactional Memory and Programming Technologies*, Armonk, March 07.
* How not to get a job in research, *Summer School on Trends in Concurrency,* Bertinoro, July 06.

* Scoped Types and Aspects for Real-Time Systems, *European Conference on Object Oriented Programming*, Nantes, July 06.
* Real-time Java in Avionics Applications. *Real-Time and Embedded Technology and Applications Symposium*, 06.
* Preemptible Atomics, *IFIP Working Group 2.4*, Jackson's Mill, West Virginia, October, 05.
* Preemptible Atomics, *IFIP Working Group 2.4* , Jackson's
* Memory Safe RTSJ Programming, *Safety & Mission Critical Workshop*, Palo Alto, September 05.
* Preemptible Atomic Regions, SUN Microsystems, August 05.
* Adopting Ownership Types, *Dagstuhl Tool for Types Workshop*, Dagstuhl, June, 05.
* Stealth Types, *Foundations of Object-Oriented Languages* panel on Extreme Typing, Long Beach, CA, January 11, 05.
* The Real-time Specification for Java: issues and opportunities, *IFIP Working Group 2.4*, Baden, January 05.
* Scoped Types for Real-time Java, *International Real-Time Systems Symposium* Lisbon, December 04.
* A semantic framework for designer transactions, *European Symposium on Programming*, Barcelona, April 04.
* Transactional Facilities for Java. *Conference on Object Oriented Programming Systems, Languages and Applications*, Vancouver, 04.
* Security and Coordination. *School on Foundations of Security Analysis and Design*, Italy, September 04.
* Real-time Java with the Ovm virtual machine. *Real-time Java Symposium*, DARPA, Arlington, July 04.
* Engineering Intermediate Representations, *IFIP Working Group 2.4*, Santa Cruz, July 03.
* Lightweight confinement for featherweight Java, *Conference on Object-Oriented Programming Systems and Languages*, San Diego, October 03.
* Subtype tests in real time. In *European Conference on Object Oriented Programming*, Darmstadt, July 03.
* Engineering a customizable intermediate representation, *Workshop on Interpreters, Virtual Machines and Emulators*, San Diego, June 03.
* Encapsulating objects with confined types, *Conference on Object-Oriented Programming Systems and Languages*, Tempa, October 01.
* Confined Types, *IFIP Working Group 2.4*, Italy, July 01.
* Confined types, *Conference on Object-Oriented Programming Systems, Languages and Applications*, Denver, October 1999.
* Efficient type inclusion tests, *Conference on Object-Oriented Programming Systems, Languages and Applications*, San Jose, October 1997.
* Near optimal hierarchical encoding of types, *European Conference on Object-Oriented Programming*, Jyvaskyla, June 1997.
* Compact dispatch tables for dynamically typed object oriented languages, *Conference on Compiler Construction*, Linkoping, Sweden, April 1996.
* Taming message passing: efficient method lookup for dynamically typed languages, *European Conference on Object-Oriented Programming*, Bologna, July 1994.
* Compile-time analysis of object-oriented programs, *Conference on Compiler Construction*, Paderborn, October 1992.

Talks at Universities and Labs

National University of Singapore (24), Huawei Research (23), INRIA (22), Charles Universisy (22), Oxford University (22), Imperial College (21), King's College (21), MIT (21), Czech Technical University (20), University of Massachusetts, Amherst (19), University of Lugano (19), University of Massachusetts, Amherst (18), Brown (17), EPFL (16), Czech Technical University (15), University of California, Irvine (14), The University of Massachusetts, Amherst (14), Technion Institute of Technology (14), Tel Aviv University (14), The Open University (14), Indiana University (14), National Science Foundation (14), Northeastern University (14), Institute of Science and Technology Austria (14). Northeastern University (13), Boston (13), Samsung (13), Facebook (13), Charles University (13), Czech Technical University, Prague (13). Google (12), Stanford (12), UIUC (12), Tsinghua, (12). INRIA Rocquencourt (11), Initiative de Recherche et Innovation sur le Logiciel Libre (11), Laboratoire d'Informatique de Paris 6 (11), Microsoft Research, Redmond (11), ETHZ (11). INRIA-Rennes (10). Imperial College (09), Microsoft Research (09), Brown University (09), EPFL (09), University of Central Florida (09). University of Lugano (08), INRIA Rocquencourt (08), INRIA Rennes (08), Ecole Polytechnique Fédérale Lausanne (08), Imperial College (08), University of California, Los Angeles (08), Edinburgh University (07), IBM T.J. Watson (07), Charles University (07), Microsoft Research (07), IBM T.J. Watson (06), Swiss Federal Institute of Technology Zurich (06), University of Bern (06), Ecole Polytechnique Fédérale, Lausanne (06), Portland State University (06), Microsoft Research (06), University of Utah (06), University of Washington (05), Carnegie Mellon University (05), University of Victoria (05), University of Alberta (05), University of Nice (03), Tokyo University (01), University of Waterloo (1999), University of Syracuse (1999), University of Pennsylvania (1999), University of Toronto (1999), University of Victoria (1999), University of Rennes (1999).

Software artifacts

I have been instrumental in the development of some open source software systems.

[1] **FastR:** A partial implementation of the R language as a runtime-specializing abstract syntax tree interpreter running on top of the JVM. Collaboration with Oracle Labs. Publications: [67], [73], [145]. Development: 2011–2014.
`github.com/allr/fastr`

[2] **JSBench:** A tool for extracting deterministic benchmark from JavaScript web pages using code instrumentation and record-replay. Collaboration with Microsoft, Mozilla and Apple. Publications: [79]. Development: 2011–2014. Press: `https://www.facebook.com/PurdueCS/posts/10151644300119116`
`http://plg.uwaterloo.ca/~dynjs/jsbench/`

[3] **Thorn:** A concurrent and distributed programming language which supports rapid software development in the style of dynamic scripting languages as well as hardening of scripts into robust programs with a gradual type system. Collaboration with IBM Research. Publications: [155], [156], [94], [88]]. Development: 2008–2012.
`http://www.thorn-lang.org`

[4] **PJAz:** The Purdue JavaScript Analyzer package is a trace-based analysis engine for JavaScript. PJAz has been used to show that common benchmarks used in the industry to measure JS performance are not representative of real-world programs and has invalidated widely held misconceptions about how the language is being used. Publications: [82], [80], [156]. Development: 2009–2011.
`http://plg.uwaterloo.ca/~dynjs/`

[5] **CDx:** A benchmark suite consisting of plain Java, real-time Java and C programs that emulate a collision detection application. CDx has been used to evaluate the performance of real-time Java virtual machines. Publications: [87], [89], [152], [151]. Development: 2004–2012.
`https://www.cs.purdue.edu/sss/projects/cdx/`

[6] **Flexotasks:** A programming model and runtime system that lets developers mix highly responsive tasks and timing-oblivious Java applications. Collaboration with IBM Research and EPFL. Publications: [97], [95], [92], [91], [28]]. Development: 2007–2009.
`http://flexotask.sourceforge.net`

[7] **StmBench7:** A benchmark for evaluating TM implementations. It aims at providing a workload that is both realistic and non-trivial to implement in a scalable way. The implementation (in Java and C++) contains a lock-based synchronization strategy that can serve as a baseline for comparison with various TMs. Collaboration with EPFL. Publications: [99]. Development: 2007–2010.
`http://lpdserver.epfl.ch/transactions/wiki/doku.php?id=stmbench7`

[8] **Ovm:** An open source framework for building virtual machines for Java-like languages. Ovm was used in the first real-time JVM deployed on a UAV. Publications: [33], [40], [96], [98], [100], [101], [103], [104], [105], [113], [120], [32], [31], [97], [102], [106]. Development: 2000–2008.
`http://janvitek.org/soft/ovm/`

[9] **MBA:** A tool for *M*odel-*B*ased protein backbone nuclear magnetic resonance *A*ssignments. Publications: [35], [38], [107], [42]. Development: 2003–2005.
`http://janvitek.org/soft/mba/`

[10] **Kacheck:** A tool for analyzing Java programs for detecting confinement violations. Kacheck has been used to analyze over 100MB of Java code. Publications: [34],[46], [47], [123], [127]. Development: 2000–2002.
`http://grothoff.org/christian/xtc/kacheck/`

[11] **JavaSeal:** A mobile agent middleware system based on Java implementing the Seal Calculus. Publications: [41], [45], [128], [**??**], [164], [196], [165], [198]. Development: 1996–1999.

[12] **Jazz:** A compression tool for Java class files. Publication: [129]. Development: 1998.

Graduated Students

[1] Julia Belyakova **PhD** NEU, "Decidable Subtyping of Existential Types for the Julia Language", 23.

[2] Artem Pelenitsyn **PhD** NEU, "Type Stability in Julia: A simple and efficeint optimization technique", 23.

[3] Benjamin Chung **PhD** NEU, "A Type System for Julia", 23.

[4] Petr Maj (CVUT) **PhD** CVUT, "Analyzing Large Code Repositories", 22.

[5] Alexi Turcotte **PhD** NEU (with Prof. F. Tip), "Optimizing Asynchronous JavaScript Applications", 23.

[6] Aviral Goel **PhD** NEU, "Data-driven ecosystem migration: Non-intrusive migration of Re ecosystem from Lazy to Strict semantics", 23.

[7] Oliver Flückiger **PhD** NEU, "Just-in-timeL Assumptions and Speculations", 22.

[8] Gregor Richards **PhD**, "Refinement of Web Software Motivated by Real-World Patterns", 14. (University of Waterloo)

[9] Filip Pizlo **PhD**, "Fragmentation tolerant real-time garbage collection", 12. (Apple)

[10] Jacques Thomas **PhD**, "Accommodative Mandatory Access Control" 11. (Amazon)

[11] Jesper H. Spring **PhD** (with Prof. Guerraoui). "Reflexes: Programming Abstractions for Highly Responsive Computing in Java", 08.

[12] Rajeev Gopalakrishna **PhD** (with Prof. Spafford). **PhD**. "Metric-driven feedback mechanism for secure software development", 06. (Intel).

[13] Bogdan Carbunar **PhD**. "Coverage Problems in Wireless Sensor Networks", 05 (U of Florida)

[14] Krzyzstof Palacz **PhD**. "Crusoe–Towards a Multicomputer Execution Environment for Java", 04. (Sun Labs).

[15] Jan Ječmen **MSc.** 24.

[16] Ming-Ho Yee 22.

[17] Anna Bolotina, 20.

[18] Jakub Zitny, **MSc**, 17. (Czech Technical Uni)

[19] Nadya Ortiz **MSc**, 12. (Apple)

[20] Fadi Meawad **MSc**, 13. (Google)

[21] Brandon Hill **MSc**, 13. (Oracle Labs)

[22] Petr Maj **MSc** 11, (Sony).

[23] Daniel Tang **MSc**, 11. (Google)

[24] Johan Östlund **MSc**, 10. (Uupsala)

[25] Jason Baker **MSc**, 07, (Google).

[26] Hiroshi Yamauchi **MSc**, 07, (Google).

[27] Christian Grothoff **MSc**, 05 (Uni of Denver).

[28] Andrey Madan **MSc**, 04, (Medtronics).

[29] Gergana Markova **MSc**, 03 (IBM)

[30] Jason M. Fox **MSc**, 03 (JPL)

[31] James Liang **MSc**, 02 (Sandia).

Current Students

[1] Sebastian Krynski (CVUT) (19)

[2] Jakob Haim (Purdue) (24)

Post-doctoral Researchers

[1] Aleksander Boruch-Gruszecki 24–

[2] Mickaël Laurent 24–

[3] Pierre Donat-Bouillud 20–

[4] Filip Krikava, 16–

[5] Tomas Kalibera 12–

[6] Alexander Kovalenko 19–23

[7] Ryan Culpepper, 17–22

[8] Konrad Siek, 16–22

[9] Paley Li, 15–18 (Oracle Research)

[10] Gustavo Petri, 12–15 (Université Paris 7)

[11] Rafal Kolanski, 13–14 (NICTA)

[12] Michal Malohlava, 12–13 (0xdata)

[13] Floreal Morandat, 11–12 (Uni. de Bordeaux)

[9] Nicholas Kidd, 09–10. (Google)

[10] Christian Hammer, 09–11 (Uni of Saarland)

[11] Ales Plsek, 09–11 (Oracle)

[12] Sylvain Lebresne, 08–09 (yakaz.com)

[13] Tomas Kalibera, 07–09 (Charles University)

[14] Tobias Wrigstad, 07–09 (Uni. of Uupsala)

[15] Antonio Cunei, 03–08 (TypeSafe)

[16] Jean Privat, 06–07 (Université du Québec)

[17] Marek Prochazka 03–05 (Euro.Space Agency)

[18] Jeremy Manson, 03–05 (Google)

[19] Michael Richmond, 02–03 (IBM Research)

Undergraduate students

Lionel Zoubritzky 18, Paul Laforgue 17, Ayaz Badouraly 17, Borja Lorente 18 (Twitter), Chakshu Goyal 18, Michal Vácha 18, Filippo Ghibellini 16, Ryan Macnack, 13 (Google). Brian Burg, 10 (University of Washington). Brett Mravec, Jason Ward, Chris Abernathy, 10. Rob Gevers, 09 (Purdue). Daniel Tang, 08 (Purdue). William Harris, 07 (University of Wisconsin-Madison). Andrew McClure, 06. Zacchary Wiggins, 05. Paul Kuliniewicz, 04. Wenchang Liu, 04 (Purdue). Filip Pizlo, 04 (Purdue). Chris Willmore, 03. Andrey Madan, 02 (Purdue). Ben Titzer, 03 (UCLA). Adam Lugowski, 02. Josh Moore, 02. Gergana Markova, 01 (Purdue). Theodore Witkamp, 03. Javed Siddique, Alen Montz, 04 (Purdue).

Internships

Ming-Ho Yee (19), Microsoft Research. Alexi Turcotte (19), Oracle Labs. Aviral Goel (19), Oracle Labs. Artem Pelenitsyn (19), Twig IO. Scott Carr (14), Microsoft Research. Fadi Meawads (13), Google. Gregor Richards (12), Oracle. Brandon Hill (12), Oracle. Fadi Meawads (12), Google. Gregor Richards (11), Mozilla. Fadi Meawad (10), Microsoft Research. Lei Zhao (10), Oracle. Gregor Richards (10), Microsoft Research. Daniel Tang (10), Google. Fadi Meawad (09), Google. Gregor Richards (09), IBM Research. Armand Navabi (09), Microsoft. Johan Östlund (09), Adobe. Jesper Spring, IBM Research. Filip Pizlo, Microsoft Resarch. Jacques Thomas, Google. Jacques Thomas, Microsoft. Krzystof Palacz, SUN Labs. Adam Welc, SUN Labs. Hiroshi Yamauchi, SUN Labs. Gergana Markova, IBM Research. Filip Pizlo, IBM Research. Christian Grothof, IBM Research. Andrew McClure, SUN Labs. Ben Titzer, SUN Labs. Andrei Madan, Medtronics.

Teaching

I regularly teach introductory courses in programming, data science, distributed programming, senior software engineering, embedded and programming languages, as well as graduate programming languages, software engineering, and embedded systems. Many of the classes where new offerings or significant overhauls of existing classes.

Fundamentals II: Introduction to Class-based Program Design. The course studies the class-based program design and the design of abstractions that support the design of reusable software and libraries. It covers the principles of object oriented program design, the basic rules of program evaluation, and examines the relationship between algorithms and data structures, as well as basic techniques for analyzing algorithm complexity. (Class designed and led by Ben Lerner)

Parallel Data Processing in MapReduce. Big data is a catchall term for datasets that are resource intensive. I redesigned this class to introduce student to parallel and distributed processing technologies for analyzing 'big data'. The course covers programming paradigms and abstractions for data analysis at scale. Students gain an understanding of the performance and usability trade-offs of various technologies and paradigms. Students will become familiar with technologies such as Hadoop, Spark, H20 and TensorFlow amongst other. Hands-on assignments focus on machine learning and data analysis tasks. The class builds on known principles such as the design recipe, testing and code reviews.

Introduction to Data Science. I created this course at Northeastern to overview the practical issues and techniques for data importing, tidying, transforming, and modeling. The course offers a gentle introduction to techniques for processing big data. Programming is a cross-cutting aspect of the course. The course work includes a term project based on real-world data. Required topics include: Data management and processing: definition and background; Data transformation; Data import; Data cleaning; Data modeling; Relational and analytic databases; Basics of SQL; Programming in R and/or Python; MapReduce fundamentals and distributed data management; Data processing pipelines, connecting multiple data management and analysis components; Interaction between the capabilities and requirements of data analysis methods (data structures, algorithms, memory requirements) and the choice of data storage and management tools; Repeatable and reproducible data analysis.

Program Design Paradigms. CS 5010 is the mandatory introductory course for students in the MS program. The course has two distinct objectives. First, it will ensure that all MS students have the same background in designing programs. This encompasses the entire design cycle, from problem analysis to the development of test suites. Second, the course will also introduce students to programming as a people discipline. Students will work in pairs, present code to panels, and learn to cope with an evolving code base. (class designed and lead by Mitch Wand)

Embedded systems. Embedded Computer Systems is Purdue's first embedded course. This course introduces students to the challenges of real-time programming and introduce Java as a general-purpose language capable of solving hard real-time problems. Low-level programs are written in C. The course covers the following topics: concurrent programming, real-time programming, worst case execution time analysis, schedulability analysis, real-time garbage collection, safety critical system verification and validation, model-driven architectures. Students performed hand on experiments in a lab configured with the RTEMS real-time operating system, LEON radiation-hardened processors, Real-time Java virtual machines, and the Rapita static program analysis tools.

Software engineering. I overhauled Purdue's SE curriculum for both graduate and undergraduate offerings. The objective of the course is to teach the principles and practical techniques needed to engineer large-scale, reliable, and secure systems. The course introduces students to a typical industrial setting where the work is in small and specialized teams, and where the projects involve composing application from off-the-shelf components rather than developing the applications from scratch. The course is based on the object-oriented approach which is particularly appropriate for achieving these goals. The material ranges from design and modeling, programming practices, refactoring, testings, design patterns and program analysis.

Programming languages. I designed this course around the book *"Programming languages, an interpreter based approach"* by Sam Kamin and Norman Ramsey and developed the additional materials and the projects. The course studies programming languages through their operational semantics and language interpreters. It gives both a formal account of the languages, and a practical feeling for implementation concerns.

Introduction to C programming. Redesigning the C programming class was a major endeavor as it entailed scaling the class from 50 students to over 280 as the department's enrollment soared. Software for automated grading had to be developed. I designed new assignments and project sequences, prepared short videos for offline learning, and used technologies such as clickers for rapid classroom feedback. I gave the class three times and my changes have been adopted by the following instructors.

Service

I enjoy helping to organize the community. I have served on over 50 conference program committees as a PC member or a chair, and about as many workshops. I have founded several successful workshop series, starting with Mobile Object Systems (**MOS**), the International Workshop on Aliasing, Confinement and Ownership (**IWACO**), the Workshop on Languages, Compilers, and Hardware Support for Transactional Computing (**TRANSACT**), the Scripts to Program Workshops (**STOP**), the International Workshop on Libraries, Languages and Compilers for Array Programming (**ARRAY**), the Machine Learning for Programming Languages Workshop (**ML4PL**), the R Implementation, Optimization and Tooling Workshop (**RIOT**), the Workshop on Speculative Side Channel Analysis (**WoSSCA**), and was the first program chair of Virtual Execution Environments (**VEE**) as well as the co-founder of the Curry On conference (**CURRYON**). Together with his colleagues Jagannathan and Grama, I founded and ran the first instances of the Summer School on Trends in Concurrency (**TiC**) which were held in Bertinoro, Prague and Bengalore and attracted over 50 PhD students each time. Together with Laurie Tratt we have created the Programming Language Implementation Summer School (**PLISS**) which was held in Bertinoro. James Noble and I started the ECOOP Summer School and the SPLASH-I tutorial series.

International Meetings and Schools Organized

[1] Curry On / REBASE conferences, *2015, 2016, 2017, 2018, 2019, 2020.*

[2] Programming Language Mentoring Workshop, OOPSLA, Boston, *2018.*

[3] Programming Language Implementation Summer School (PLISS), Bertinoro, *2017, 2019, 2022.*

[4] Dagstuhl Seminar on *Rethinking Experimental Methods in Computing*, *March 2016.*

[5] ECOOP Summer School, *2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019.*

[6] SPLASH-I Tutorial Series, *2013, 2015, 2018.*

[7] The State of the Union JS (SOTU.JS), San Jose, *April 2014.*

[8] NSF DALI Workshop on Dynamic Languages for Scalable Data Analytics, Indianapolis, *2013.*

[9] NSF Workshop on Programming with Big Data, Hawaii, *2013.*

[10] Dagstuhl Seminar on *Foundations for Scripting Languages*, *February 2012.*

[11] Virtual Execution Environments for Scientific Computing NSF Workshop, Arlington, *September 2010.*

[12] IFIP WG 2.4 Working group meeting. Bormio, Italy, 2009

[13] Dagstuhl Seminar on *Types for Tools: Applications of Type Theoretic Techniques June 2005.*

[14] International Summer School on Trends in Concurrency, *July 2006, 2008, 2010.*

Comfy Chair

[1] European Joint Conferences on Theory and Practice of Software (ETAPS), Prague, 2019.

[2] European Conference on Object Oriented Programming, Amsterdam, *July 2018.*

[3] European Conference on Object Oriented Programming, Barcelona, *July 2017.*

[4] European Conference on Object Oriented Programming, Rome, *July 2016.*

[5] European Conference on Object Oriented Programming, Prague, *July 2015.*

[6] SIGPLAN SPLASH Conference, Indianapolis, *October 2013.*

General Chair

[1] Conference on Systems, Programming, Languages and Applications: Software for Humanity (**SPLASH**), *November 2018.*

[2] Curry On Conference, (**CurryOn**), *July 2015.*

[3] Conference on Languages, Programming Language Design and Implementation (**PLDI**), *June 2012.*

[4] Conference on Languages, Compilers and Tools for Embedded Systems (**LCTES**), *June 2011.*

[5] International Memory Management Symposium (**ISMM**), *2010.*

[6] Workshop on Languages, Compilers, & Hw Support for Transactional Computing (**TRANSACT**), *2006.*

Program Chair

[1] Conference on Object-Oriented Programming Languages, Systems and Applications (**OOPSLA**), 2022.

[2] European Conference on Object Oriented Programming (**ECOOP**) 2008, 2022.

[3] Artifact Evaluation Committee, *OOSPLA 2018, OOPSLA 2019, POPL 2015, ECOOP 2013, PLDI 2014.*

[4] European Symposium on Programming (**ESOP**) 2015.

[5] Conference on Objects, Models, Components, Patterns (**TOOLS Europe**) 2010.

[6] Java Technologies for Real-time and Embedded Systems (**JTRes**) symposium 2010.

[7] European Conference on Object Oriented Programming (**ECOOP**) 2008.

[8] Conference on Coordination Models and Languages (**COORDINATION**) 2007.

[9] Conference on Virtual Execution Environments (**VEE**) 2005.

[10] Formal Techniques for Java-like Programs (**FTfJP**) workshop 2005.

[11] Java Technologies for Real-time and Embedded Systems (**JTRes**) workshop, 2005.

Conference Program Committees

* **ACSD** – International Conference on Application of Concurrency to System Design, *2012.*
* **AISB** – Symposium on Software Mobility and Adaptive Behavior, *2001.*
* **ASA/MA** – Agent Systems and Applications, Mobile Agents, *2001.*
* **APLAS** – Asian Symposium on Programming Languages and Systems, *2012, 2014.*
* **CATS** – Computing: The Australasian Theory Symposium, *2010.*
* **CC** – International Conference on Compiler Construction, *2003, 2008, 2012, 2014, 2021.*
* **CD** – Component Deployment, *2002, 2004.*
* **COORD** – International Conference on Coordination Models and Languages, *2005, 2008, 2009.*
* **CSF** – IEEE Computer Security Foundations Symposium, *2008.*

* **DATE** – Design, Automation & Test in Europe, Conference, *2010*.
* **DLS** – Dynamic Language Symposium Conference, *2010, 2014, 2015*.
* **ECOOP** – European Conference on Object Oriented Programming, *1998, 2000, 2001, 2002, 2003, 2007, 2008, 2009, 2010, 2013, 2020*.
* **ESOP** – European Symposium on Programming, *2002, 2007, 2009, 2011, 2014*.
* **EUC** – IEEE/IFIP International Conference On Embedded and Ubiquitous Computing, *2008, 2010*.
* **GPCE** – Conference on Generative Programming: Concepts & Experiences, *2013*
* **HVC** – Haifa Verification Conference, *2014*.
* **HOTPAR** – USENIX Hot Topics in Parallelism, *2013*.
* **ICFP** – ACM SIGPLAN International Conference on Functional Programming, *2005*.
* **ICALP** – International Conference on Automata, Languages and Programming, *2000*.
* **ISMM** – International Symposium on Memory Management, *2010*.
* **ISORC** – International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, *2012*.
* **JFLA** – Journées Francophones des Langages Applicatifs, *1995, 1998, 2000*.
* **MASS** – Symposium on Multi-Agent Security and Survivability, *2004, 2005*.
* **PLDI** – Conference on Programming Language Design and Implementation, *2001, 2010, 2013*.
* **PPPJ** – International conference on Principles and Practice of Programming in Java,*2006*.
* **PODC** – Symposium on Principles of Distributed Computing, *2010*.
* **POPL** – ACM SIGPLAN Conference on Principles of Programming Languages, *2001, 2007, 2011*.
* **RTSS** – IEEE International Real-Time Systems Symposium, *2009, 2010, 2011*.
* **SACMAT** – Symposium on Access Control Models and Technologies, *2001, 2008*.
* **SNAPL** – Summit on Advances in Programming Languages, *2015*.
* **TOOLS** – TOOLS Europe, *2011, 2019*.
* **OOPS** – Object Oriented Programming Languages and Systems *2004, 2005*.
* **OOPSLA** – ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications, *2000, 2004, 2007, 2008, 2023*.
* **VEE** – Virtual Execution Environments, *2011*.

Workshop Program Committees

AcadPub – International Workshop on Academic Publishing 2.0, *2014*. **AIOOL** – International Workshop on Abstract Interpretation of Object-Oriented Languages, *2005*. **ACP4IS** – Workshop on Aspects, Components, and Patterns for Infrastructure Software, *2003, 2004*. **ARRAY** – Workshop on Libraries, Languages, and Compilers for Array Programming, *2014*, *2015*. **Bytecode** – Workshop on Bytecode Semantics, Verification Analysis and Transformation, *2007, 2008*. **CORD** – Workshop on Concurrency, Real-Time and Distribution in Eiffel (CORDIE), *2006*. **CPS** – International Workshop on Cyber-Physical Systems, *2008*. **CSJP** – Workshop on Concurrency and Synchronization in Java Programs, *2004*. **DDFP** – Data Driven Functional Programming Workshop, *2013*. **DOSW** – Distributed Object Security Workshop. *1999*. **Euro-Par** – Euro-Par 2014 Workshops, *2014*. **Express** – International Workshop on Expressiveness in Concurrency, *2011*. **FOCLASA** – International Workshop on the Foundations of Coordination Languages and Software Architectures, *2004, 2005, 2007*. **FOOL** – Workshop on Foundations of Object-Oriented Languages, *2013*. **GCM** – Green Computing Middleware, *2010*, *2011, 2012*. **ICOOO** – Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems. (ICOOOLPS), *2006, 2013, 2014, 2015*. **IWACO** – International Workshop on Aliasing, Confinement and Ownership, *2003, 2007*, 2014. **IWMSE** – Third International Workshop on Multicore Software Engineering, *2010*. **IWAOOS** – Intercontinental Workshop on Aliasing in Object-Oriented Systems. *1999*. **JTRes** – Workshop on Java Technologies for Real-Time and Embedded Systems, *2003, 2004, 2005, 2006, 2007, 2008, 2009,*, *2010*, *2011*, *2012, 2013, 2014, 2015*. **ML4PL** – Machine Learning for Programming Languages, *2015*. **MOS** – Mobile Object Systems Workshop, *1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002 2003, 2004, 2005*. **OBT** – Off-the Beaten Track, *2014* **PLACES** – Programming Language Approaches to Concurrency and Communication-cEntric Software, *2009*, *2010*, *2011*, *2012*. **PLAS** – Workshop on Programming Languages Analysis for Security,

2007 **PLASTIC** – Workshop on Programming Language And Systems Technologies for Internet Clients, *2011*. **PSW** – Workshop on Programming the Semantic Web, *2012*. **PWD** – Workshop on Program Analysis for Security and Safety (PASSWORD), *2006* **Scala** – Annual Scala Symposium, *2015*. **SALAD** – Workshop on SoftwAre debLoating And Delayering, *2018*. **SecCo** – International Workshop on Security Issues in Coordination Models, Languages and Systems, *2003, 2004, 2005, 2007*. **RIOT** – R Implementation, Optimization and Tooling Workshop, *2015, 2019*. **STOP** – Script to Program Evolution, *2009*. **TRUST** – Workshop on Reproducible Research Methodologies and New Publication Models, *2014*. **VMIL** – Workshop on Virtual Machines and Intermediate Languages, *2009*. **WASD** – Workshop on Advanced/Academic Software Development Tools and Techniques, *2013*. **WSIC** – Workshop on Secure Internet Computations. Organizer *1999*. **WoSSCA** – Workshop on Speculative Side Channel Analysis, *2018*

Department and School service

At Purdue University:

- Graduate admissions Chair: *11 – 14*.
- Graduate admissions: *04 – 05*, *07 – 11*.
- Hiring committee: *09, 14*.
- Colloquium chair: *03, 04*.
- Graduate committee: *99 – 02*.
- Student appeal committee: *04 – 05*, *10 – 12*.

At Northeastern University

- Graduate committee: *2014*.
- Graduate admission: *2015, 2017, 2018, 2019, 2022.*
- Hiring committee: *2016*.

Grants

[1] PI (100%) *SHFL Small: Predictable Performance for Just-in-Time Compilation*, **NSF**, $500,000, 22 – 25.

[2] PI (70%) *CCRI: ENS: Collaborative Research: Enhancing R for Scalability and Deployment to Support Scientific Communities*, **NSF**, $1,791,794, 19 – 22.

[3] PI (100%) *SHF: Small: Program Analysis for Data Science*, **NSF**, $499,671, 19 – 22.

[4] Co-PI (50%) *SHF: Small: Collaborative Research: A Rational Reconstruction of the Julia Type System*, **NSF**, $494,444, 19 – 22.

[5] Co-PI (40%) *ABI Innovation: Scalable and Agile Analysis of Mass Spectrometry Experiments*, **NSF**, $791,794, 18 – 21.

[6] PI (30%) *VerticA: Towards Integrated, Trustworthy, Scripting Languages*, **ONR**, $1,596,197, 17 – 20.

[7] PI *BigCode*, **Czech Operational Programme Research, Development, and Education**, €1,890,000, 18 – 22.

[8] PI *Evolving Language Ecosystems (ELE)*, **ERC Advanced**, €3,200,000 , 16 – 22.

[9] Co-PI (15%) *SHF: Large Gradual Typing Across the Spectrum*, **NSF**, $1,600,000, 15 – 19.

[10] PI *SHF: Project Darwin*, **NSF**, $1,099,000, 16 – 20.

[11] PI (50%) *SHF: Small: Foundations of Just-in-Time Compilation*, **NSF**, $436,000, 16 – 19.

[12] Co-PI (50%) *SHF: Small Havoc: Verified Compiliation of Concurrent Managed Languages*, **NSF**, $400,000, 15 – 18.

[13] PI *JavaScript Analysis*, **Google**, $64,000, 16 –.

[14] PI *HyDyS: Hybrid Dynamic/Static Analysis for Managed Languages*, **Google**, $60,000, 14.

[15] PI *FastR: Open Source R*, **Oracle Inc**, $150,000, 13.

[16] PI *Hybrid Dynamic and Static Techniques for Trustworthy Data Analytics*, **ONR**, $600,000, 13 – 16.

[17] Co-PI (50%) *Verified Compilation of Concurrent Managed Languages*, **AFRL**, $300,000, 13 – 16.

[18] PI *CSR: Language and Runtime Support for Large-Scale Data Analytics*, **NSF**, $246,635, 12.

[19] PI *FastR: Open Source R*, **Oracle Inc**, $85,000, 12.

[20] PI *Automated Generation of JavaScript Workloads*, **Mozilla Corporation**, $75,000, 12.

[21] PI (25%) *CPS: Robust Distributed Wind Power Engineering*, **NSF**, $1,600,000, 11 – 15.

[22] PI (50%) *SI2-SSE: A Tracing Virtual Machine For Statistical Computing*, **NSF**, $489,084, 10 – 13.

[23] PI *EAGER: Foundations of Data-Centric Concurrency Control*, **NSF**, $110,000, 10 – 11.

[24] PI *Virtual Execution Environments for Scientific Computing Workshop*, **NSF**, $45,000, 10 – 10.

[25] PI *Third International Summer School on Trends in Concurrency*, **NSF**, $12,000, 10 – 10.

[26] Co-PI (50%) *An Infrastructure for Scalable Transactional Memory Abstractions*, **NSF**, $536,000, 10.

[27] Co-PI (50%) *SHF: Specification and Verification of Safety Critical Java*, **NSF**, $500,000, 09–11.

[28] PI *A Computational Model for High-Assurance Dynamic Systems*, **ONR**, $200,000, 09– 09.

[29] PI (50%) *Certified Garbage Collection for Highly Responsive Systems*, **NSF**, $498,952, 08–11.

[30] Co-PI (50%) *Fault Determination and Recovery in Cycle Sharing Infrastructures*, **NSF**, $23,000, 08.

[31] PI (25%) *Unified Open Source Transactional Infrastructure*, **NSF**, $1,000,000, 08 – 11.

[32] Co-PI *Language & Runtime Support for Safe and Scalable Programs*, **Microsoft Research**, $50'000 (25%), 08.

[33] PI *Second International Summer School: Trends in Concurrency*, **NSF**, $23,000, 08.

[34] PI *Second International Summer School: Trends in Concurrency*, **IBM Research**, $1,000, 08.

[35] PI *Second International Summer School: Trends in Concurrency*, **Microsoft Research**, $10,000, 08.

[36] PI *Second International Summer School: Trends in Concurrency*, **Intel Research**, $5,000, 07.

[37] PI *CSR-EHS: High-throughput Real-time Stream Processing in Java*, **NSF**, $210,000, 07 – 10.

[38] Co-PI *Controlled Declassification with Software Transactional Memory*, **NSF**, $249,857 (50%), 07 – 09.

[39] PI *IBM Faculty Award*, **IBM**, $30,000, 06.

[40] PI *High-level Concurrency Control Abstractions* , **Microsoft Research Award**, $50,000, 06.

[41] Co-PI *An Infrastructure for Relaxed Concurrency Abstractions*, **NSF**, $99,979 (20%), 06 – 08.

[42] PI *International Summer School: Trends in Concurrency*, **Microsoft Research**, $5,000, 06.

[43] PI *International Summer School: Trends in Concurrency*, **IBM Italy**, $5,000, 06.

[44] Co-PI *Fault Determination and Recovery in Cycle-Sharing Infrastructures*, **NSF**, $300,000, 05 – 08.

[45] PI *Aspectual Configuration of Real-time Embedded Middleware*, **NSF**, $250,000, 05 – 08.

[46] Co-PI *A logically destructive imaging security & forensics facility*, **NSF**, $800,000 (7%), 04 – 07.

[47] PI *Assured Software Composition For Real-Time Systems*, **NSF/NASA**, $500,000, 03 – 07.

[48] PI *Language Abstractions for Parallel Computing*, **DARPA PERCS**, $400,000, 03 – 06.

[49] Co-PI *Partage: An Open Peer-to-Peer Infrastructure for Cycle-Sharing*, **NSF**, $498,945, 03 – 06.

[50] Co-PI *Distributed Access Control for Accountable Systems*, **NSF Cybertrust**, $318,375, 02 – 06.

[51] PI *Foundations & Implementation of Mobile Object Systems*, **NSF CAREER**, $325,936, 01 – 06.

[52] PI *Dynamic Compositional Middleware Frameworks for Networked Embedded Systems*, **DARPA**, $3,274,680 (60%), 01 – 05.

[53] PI *Sw. Eng.: Research on Customizable Virtual Machines*, **Microsoft Research**, $100,000, 02.

[54] PI *Trusted Software Composition*, **Eli Lilly**, $50,000, 01 – 02.

[55] PI *ReAssure–Secure and Resilient Network Computing*, **Eli Lilly**, $90,000, 99 – 01.

[56] PI *Resilient Mobile Agent Architecture*, **Motorola**, $62,543, 00 – 05.

[57] PI *Type confinement in Java*, **Eli Lilly**, $25,000, 99 – 00.