Final Report for the Emmy Noether Project

ConcSys: Reliable and Efficient Complex, Concurrent Software Systems

Michael Pradel
University of Stuttgart

1 General Information

DFG Geschäftszeichen	PR 1537/1-1
Head of project	Michael Pradel
Institution	Department of Computer Science University of Stuttgart
Topic	ConcSys: Reliable and Efficient Complex, Concurrent Software Systems
Period reported about	March 2015 – December 2024
Appointment to a permanent professorship in the course of the funding	Yes

2 Summary

2.1 English Version

The ConcSys project aims to develop techniques for testing and analyzing complex software systems, with a focus on increasing the correctness and performance of such systems. The project was running from March 2015 until December 2024. In this period, we made significant progress, both in terms of scientific results and in terms of building up a research group. The scientific results include novel techniques for (i) finding and preventing concurrency bugs, (ii) understanding and analyzing software performance, (iii) automated test generation, (iv) program analysis for WebAssembly, and (v) foundations of dynamic analysis. These results are presented in 83 peer-reviewed publications at top-tier conferences and journals in software engineering and programming languages, e.g., ICSE, OOPSLA, PLDI, and FSE. Beyond these scientific results, the project has enabled the PI, Michael Pradel, to build up his own a research group, to establish himself as an internationally recognized leader in the field, and to secure a permanent professorship at the University of Stuttgart. The project has directly and indirectly contributed to the careers of 12 doctoral students, out of which seven have been partially funded by the project and six have already graduated.

2.2 German Version

Das ConcSys-Projekt hat zum Ziel, Techniken zum Testen und Analysieren komplexer Softwaresysteme zu entwickeln, mit einem Schwerpunkt auf der Erhöhung der Korrektheit und Leistungsfähigkeit solcher Systeme. Das Projekt lief von März 2015 bis Dezember 2024. In diesem Zeitraum haben wir bedeutende Fortschritte erzielt, sowohl in wissenschaftlicher Hinsicht als auch beim Aufbau einer Forschungsgruppe. Zu den wissenschaftlichen Ergebnissen gehören neuartige Techniken für (i) das Finden und Verhindern von Nebenläufigkeitsfehlern, (ii) das Verständnis und die Analyse der Software-Effizienz, (iii) die automatische Testgenerierung, (iv) die Programmanalyse für WebAssembly und (v) die Grundlagen der dynamischen Analyse. Diese Ergebnisse wurden in 83 von Fachkollegen/innen begutachteten Veröffentlichungen auf erstklassigen Konferenzen und in renommierten Zeitschriften im Bereich Software Engineering und Programmiersprachen präsentiert, z.B. ICSE, OOPSLA, PLDI und FSE. Über diese wissenschaftlichen Ergebnisse hinaus hat das Projekt dem Projektleiter, Michael Pradel, ermöglicht, eine eigene Forschungsgruppe aufzubauen, sich als international anerkannte Führungspersönlichkeit auf diesem Gebiet zu etablieren und eine W3 Professur an der Universität Stuttgart zu erhalten. Das Projekt hat direkt und indirekt zur Karriere von 12 Doktorandinnen und

Doktoranden beigetragen, von denen sieben teilweise durch das Projekt finanziert wurden und sechs bereits promoviert haben.

3 Scientific Progress Report

The project has achieved many of the goals outlined in the proposal. Moreover, we have expanded the original scope of the project to other problems related to the reliability and efficiency of both sequential and concurrent complex software systems. The following summarizes the main results of the project in five orthogonal fields of research: (i) finding and preventing concurrency bugs, (ii) understanding and analyzing software performance, (iii) automated test generation, (iv) program analysis for WebAssembly, and (v) foundations of dynamic analysis.

3.1 Finding and Preventing Concurrency Bugs

We made several contributions to the detection and prevention of concurrency bugs in software systems. The following highlights three of these contributions.

First, CovCon, introduced in an ICSE 2017 paper, is a coverage-guided approach for automatically generating concurrent tests. Unlike random or pattern-specific methods, CovCon strategically focuses on method pairs that have not been executed concurrently, making it both general and efficient. Applied to 18 thread-safe Java classes, it identified bugs in 17, outperforming five state-of-the-art techniques in both effectiveness and speed, with notable speedups in many cases.

Second, our PLDI 2017 paper introduces Simian, a fully automated system for analyzing multi-client web applications. Using a two-phase black-box strategy, Simian first identifies potential client-side conflicts through single-client exploration and then synthesizes multi-client interactions to expose inconsistencies. It successfully reveals bugs in widely-used systems like Google Docs and Firepad, while running 10x faster than exhaustive methods.

Finally, our ASE 2018 paper presents TSFinder, a tool that automatically infers thread safety documentation using a graph-based classifier trained on known thread-safe and unsafe classes. TSFinder achieves 94.5% accuracy and processes each class in about three seconds, making it a practical solution for improving documentation in concurrent software.

3.2 Understanding and Analyzing Software Performance

To address the challenge of understanding and optimizing software performance, we have developed several techniques that focus on different aspects of performance analysis, especially in concurrent programs and JavaScript-based applications.

SyncProf, introduced in ISSTA 2016, is a concurrency-focused profiler that identifies synchronization bottlenecks in C/C++ programs. It uses repeated executions and a novel graph-based representation of critical sections to pinpoint root causes and recommend optimizations. SyncProf outperforms traditional lock contention profilers in both accuracy and usefulness.

PerfSyn (CGO 2018) introduces a method for synthesizing test programs that expose performance bottlenecks in Java methods. By modeling the synthesis task as a combinatorial search, PerfSyn automatically discovers inefficiencies, such as unexpected computational complexity, in a wide range of software components.

In the context of more general optimization, DecisionProf (ISSTA 2017) offers actionable profiling by analyzing logical expressions and switch statements in JavaScript programs to suggest more efficient evaluation orders. This dynamic analysis proposes code changes only when statistically significant performance improvements are likely, achieving speedups of up to 59% in individual functions across 43 real-world projects.

Lastly, our empirical study presented at ICSE 2016 investigates 98 fixed performance issues across 16 JavaScript projects. It identifies common root causes–most notably inefficient API usage–and shows that many issues are resolved through small, low-complexity code changes. Moreover, the study finds recurring optimization patterns and reveals that performance improvements are not always consistent across JavaScript engine versions, offering valuable insights for developers and researchers alike.

3.3 Automated Test Generation

The ability to automatically generate tests is crucial for ensuring the correctness of software systems, especially in the context of complex and concurrent applications. We highlight several of our contributions in this area in the following, which address important limitations of existing techniques across a variety of software domains.

LambdaTester (OOPSLA 2018) targets the challenge of testing higher-order functions in dynamic languages like JavaScript. It automatically infers callback positions and generates meaningful callback functions to trigger otherwise untested behaviors, significantly increases the coverage of the tested JavaScript libraries.

Similarly, TestMiner (ASE 2017) improves test generation for domain-specific Java software by mining input values from a large corpus of existing tests. This enables more effective testing of hard-to-reach code and increases test coverage by 21% compared to existing tools.

Other approaches focus on generating tests for more complex environments and behaviors. FuzzDroid (ICSE 2017) is a framework for automatically creating Android environments to expose malicious app behavior. By combining static and dynamic analysis in a search-based strategy, FuzzDroid reaches target malicious code locations in 75% of recent malware samples, often within a minute.

At the UI level, a test generation technique introduced in ISSTA 2016 leverages human execution traces to infer "macro events"–logical interaction steps that consist of multiple low-level UI actions. Applied to real-world web apps, this approach significantly improves coverage and realism compared to random testing, validating more functionality within the same time budget.

3.4 Program Analysis for WebAssembly

WebAssembly, a relatively new language that was introduced in 2017, is supported by all major web browsers and several other execution platforms beyond the web. From a program analysis perspective, it is interesting because it offers new challenges, such as a stack-based virtual machine, and new opportunities, such as a well-defined type system and clearly defined formal semantics. We contributed to WebAssembly analysis by developing techniques to enhance its correctness, security, and a framework for general-purpose analysis.

One example is the Wasabi framework, which received the *Best Paper Award* at ASPLOS 2019. Wasabi is a general-purpose dynamic analysis framework for WebAssembly that allows developers to easily implement and evaluate new analyses. It provides a set of APIs for instrumenting WebAssembly code, collecting runtime information, and analyzing the execution behavior.

Another example is our study on binary security problems in WebAssembly, which was published at USENIX Security 2020. We analyzed the security implications of WebAssembly's design and implementation, identifying potential vulnerabilities and proposing mitigations. We also conducted an empirical study of real-world WebAssembly binaries, published at WWW 2021.

3.5 Foundations of Dynamic Analysis

One promising way to reason about the correctness and performance of software systems is to observe their execution behavior. By its nature, dynamic analysis assumes that we can execute the software under analysis. However, executing complex software systems in a way that reaches a specific code location to analyze often requires significant effort, e.g., because the system is difficult to set up and the target location is difficult to reach.

To address this problem, we have developed a new approach called "learning-guided execution", presented at FSE 2023 and awarded with an *ACM/SIGSOFT Distinguished Paper Award*. The approach combines machine learning and program instrumentation to guide the execution of software systems towards specific code locations of interest.

Based on the general concept of learning-guided execution, we also explored its applications. One such application is presented in ChangeGuard, to be presented at FSE 2025. This approach validates code changes meant to be refactorings by executing the old and the new code side-by-side. To this end, we introduce pairwise learning-guided execution, which extends the original idea of learning-guided execution to pairs of simultaneously executed program.

We also further improved upon our LExecutor work by introducing an LLM-based, feedback-directed version of this idea. It is called “TreeFix” and will be presented at ICSE 2025.

4 Published Project Results

4.1 Peer-Reviewed Publications

In the course of the project, the PI and his collaborators have published 83 peer-reviewed publications at top-tier conferences and journals in software engineering and programming languages, e.g., ICSE, OOPSLA, PLDI, and FSE. The complete list of publications is given in the appendix.

4.2 Other Published Results

Following the vision of open science, almost all of the publications listed above come with a publicly released artifact. These artifacts contain the implementation of the technique described in the paper, datasets gathered during the evaluation, and a set of scripts to reproduce the results. See `http://software-lab.org/publications.html` for a full list of publications and their artifacts.

4.3 Patents

None

Appendix: Peer-Reviewed Publications

FSE'25 **ChangeGuard: Validating Code Changes via Pairwise Learning-Guided Execution**
Lars Gröninger, Beatriz Souza, Michael Pradel. *International Conference on the Foundations of Software Engineering*

FSE'25 **An Empirical Study of Suppressed Static Analysis Warnings**
Huimin Hu, Yingying Wang, Julia Rubin, Michael Pradel. *International Conference on the Foundations of Software Engineering*

ICSE'25 **RepairAgent: An Autonomous, LLM-Based Agent for Program Repair**
Islem Bouzenia, Premkumar Devanbu, Michael Pradel. *International Conference on Software Engineering*

ICSE'25 **Treefix: Enabling Execution with a Tree of Prefixes**
Beatriz Souza, Michael Pradel. *International Conference on Software Engineering*

ICSE'25 **Calibration and Correctness of Language Models for Code**
Claudio Spiess, David Gros, Kunal Suresh Pai, Michael Pradel, Md Rafiqul Islam Rabin, Amin Alipour, Susmit Jha, Premkumar Devanbu, Toufique Ahmed. *International Conference on Software Engineering*

MSR'25 **Can LLMs Replace Manual Annotation of Software Engineering Artifacts?**
Toufique Ahmed, Premkumar Devanbu, Christoph Treude, Michael Pradel. *International Conference on Mining Software Repositories*

OOPSLA'24 **Wasm-R3: Record-Reduce-Replay for Realistic and Standalone WebAssembly Benchmarks**
Doehyun Baek, Jakob Getz, Yusung Sim, Daniel Lehmann, Ben Titzer, Sukyoung Ryu, Michael Pradel. *Proceedings of the ACM on Programming Languages*

FSE'24 **Analyzing Quantum Programs with LintQ: A Static Analysis Framework for Qiskit**
Matteo Paltenghi, Michael Pradel. *International Conference on the Foundations of Software Engineering*

FSE'24 **DyPyBench: A Benchmark of Executable Python Software**
Islem Bouzenia, Bajaj Piyush Krishan, Michael Pradel. *International Conference on the Foundations of Software Engineering*

ICSE'24 **PyTy: Repairing Static Type Errors in Python**
Yiu Wai Chow, Luca Di Grazia, Michael Pradel. *International Conference on Software Engineering*

ICSE'24 **Fuzz4All: Universal Fuzzing with Large Language Models**
Chunqiu Steven Xia, Matteo Paltenghi, Jia Le Tian, Michael Pradel, Lingming Zhang. *International Conference on Software Engineering*

ICSE'24 **Resource Usage and Optimization Opportunities in Workflows of GitHub Actions**
Islem Bouzenia, Michael Pradel. *International Conference on Software Engineering*

FSE'23 **LExecutor: Learning-Guided Execution**
Beatriz Souza, Michael Pradel. *Symposium on the Foundations of Software Engineering*

ISSTA'23 **That's a Tough Call: Studying the Challenges of Call Graph Construction for WebAssembly**
Daniel Lehmann, Michelle Thalakottur, Frank Tip, Michael Pradel. *International Symposium on Software Testing and Analysis*

ISSTA'23 **Beware of the Unexpected: Bimodal Taint Analysis**
Yiu Wai Chow, Max Schäfer, Michael Pradel. *International Symposium on Software Testing and Analysis*

ICSE'23 **MorphQ: Metamorphic Testing of the Qiskit Quantum Computing Platform**
Matteo Paltenghi, Michael Pradel. *International Conference on Software Engineering*

ICSE'23 **When to Say What: Learning to Find Condition-Message Inconsistencies**
Islem Bouzenia, Michael Pradel. *International Conference on Software Engineering*

ICSE'23 **SecBench.js: An Executable Security Benchmark Suite for Server-Side JavaScript**
Masudul Hasan Masud Bhuiyan, Adithya Srinivas Parthasarathy, Nikos Vasilakis, Michael Pradel, Cristian-Alexandru Staicu. *International Conference on Software Engineering*

ICSE'23 **VulGen: Realistic Vulnerability Generation Via Pattern Mining and Deep Learning**
Yu Nong, Yuzhe Ou, Michael Pradel, Feng Chan, Haipeng Cai. *International Conference on Software Engineering*

TSE'22 **DiffSearch: A Scalable and Precise Search Engine for Code Changes**
Luca Di Grazia, Paul Bredl, Michael Pradel. *IEEE Transactions on Software Engineering*

CSUR'22 **Code Search: A Survey of Techniques for Finding Code**
Luca Di Grazia, Michael Pradel. *ACM Computing Surveys*

FSE'22 **DynaPyt: A Dynamic Analysis Framework for Python**
Aryaz Eghbali, Michael Pradel. *Symposium on the Foundations of Software Engineering*

FSE'22 **The Evolution of Type Annotations in Python: An Empirical Study**
Luca Di Grazia, Michael Pradel . *Symposium on the Foundations of Software Engineering*

FSE'22 **Generating Realistic Vulnerabilities via Neural Code Editing: An Empirical Study**
Yu Nong, Yuzhe Ou, Michael Pradel, Feng Chen, Haipeng Cai. *Symposium on the Foundations of Software Engineering*

ASE'22 **CrystalBLEU: Precisely and Efficiently Measuring the Similarity of Code**
Aryaz Eghbali, Michael Pradel. *International Conference on Automated Software Engineering*

OOPSLA'22 **Bugs in Quantum Computing Platforms: An Empirical Study**
Matteo Paltenghi, Michael Pradel. *Proceedings of the ACM on Programming Languages*

PLDI'22 **Finding the Dwarf: Recovering Precise Types from WebAssembly Binaries**
Daniel Lehmann, Michael Pradel. *Conference on Programming Language Design and Implementation*

ICSE'22 **Nalin: Learning from Runtime Behavior to Find Name-Value Inconsistencies in Jupyter Notebooks**
Jibesh Patra, Michael Pradel. *International Conference on Software Engineering*

ICSE'22 **Nessie: Automatically Testing JavaScript APIs with Asynchronous Callbacks**
Ellen Arteca, Sebastian Harner, Michael Pradel, Frank Tip. *International Conference on Software Engineering*

S&P'22 **Wobfuscator: Obfuscating JavaScript Malware via Opportunistic Translation to WebAssembly**
Alan Romano, Daniel Lehmann, Michael Pradel, Weihang Wang. *Symposium on Security and Privacy*

CACM'22 **Neural Software Analysis**
Michael Pradel, Satish Chandra. *Communications of the ACM* 65(1), pages 86–96

ASE'21 **Thinking Like a Developer? Comparing the Attention of Humans with Neural Models of Code**
Matteo Paltenghi, Michael Pradel. *International Conference on Automated Software Engineering*

FSE'21 **Semantic Bug Seeding: A Learning-Based Approach for Creating Realistic Bugs**
Jibesh Patra, Michael Pradel. *Symposium on the Foundations of Software Engineering*

CCS'21 **Preventing Dynamic Library Compromise on Node.js via RWX-Based Privilege Reduction**
Nikos Vasilakis, Cristian-Alexandru Staicu, Grigoris Ntousakis, Konstantinos Kallas, Ben Karel, Andre DeHon, Michael Pradel. *Conference on Computer and Communications Security*

ISSTA'21 **Finding Data Compatibility Bugs with JSON Subschema Checking**
Andrew Habib, Avraham Shinnar, Martin Hirzel, Michael Pradel. *International Symposium on Software Testing and Analysis*

ISSTA'21 **Continuous Test Suite Failure Prediction**
Cong Pan, Michael Pradel. *International Symposium on Software Testing and Analysis*

ICSE'21 **IdBench: Evaluating Semantic Representations of Identifier Names in Source Code**
Yaza Wainakh, Moiz Rauf, Michael Pradel. *International Conference on Software Engineering*

WWW'21 **An Empirical Study of Real-World WebAssembly Binaries: Security, Languages, Use Cases**
Aaron Hilbig, Daniel Lehmann, Michael Pradel. *The Web Conference (WWW)*

ICPE'21 **ConfProf: White-Box Performance Profiling of Configuration Options**
Xue Han, Tingting Yu, Michael Pradel. *International Conference on Performance Engineering*

IEEE Sw.'21 **Automatic Program Repair**
Claire Le Goues, Michael Pradel, Abhik Roychoudhury, Satish Chandra. *IEEE Software* 38(4), pages 22–27

ASE'20 **No Strings Attached: An Empirical Study of String-related Software Bugs**
Aryaz Eghbali, Michael Pradel. *International Conference on Automated Software Engineering*

USENIX Security'20 **Everything Old is New Again: Binary Security of WebAssembly**
Daniel Lehmann, Johannes Kinder, Michael Pradel. *USENIX Security Symposium*

FSE'20 **TypeWriter: Neural Type Prediction with Search-based Validation**
Michael Pradel, Georgios Gousios, Jason Liu, Satish Chandra. *Symposium on the Foundations of Software Engineering*

ISSTA'20 **Scaffle: Bug Localization on Millions of Files**
Michael Pradel, Vijayaraghavan Murali, Rebecca Qian, Mateusz Machalica, Erik Meijer, Satish Chandra. *International Symposium on Software Testing and Analysis*

ICSE'20 **Extracting Taint Specifications for JavaScript Libraries**
Cristian-Alexandru Staicu, Martin Toldam Torp, Max Schäfer, Anders Møller, Michael Pradel. *International Conference on Software Engineering*

IEEE Sw.'20 **Satisfying Increasing Performance Requirements with Caching at the Application Level**
Jhonny Mertz, Ingrid Nunes, Luca Della Toffola, Marija Selakovic, Michael Pradel. *IEEE Software*

OOPSLA'19 **Getafix: Learning to Fix Bugs Automatically**
Johannes Bader, Andrew Scott, Michael Pradel, Satish Chandra. *Conference on Object-Oriented Programming, Systems, Languages, and Applications*

ISSTA'19 **Interactive Metamorphic Testing of Debuggers**
Sandro Tolksdorf, Daniel Lehmann, Michael Pradel. *International Symposium on Software Testing and Analysis*

USENIX Security'19 **Small World with High Risks: A Study of Security Threats in the npm Ecosystem**
Markus Zimmermann, Cristian-Alexandru Staicu, Cam Tenny, Michael Pradel. *USENIX Security Symposium*

USENIX Security'19 **Leaky Images: Targeted Privacy Attacks in the Web**
Cristian-Alexandru Staicu, Michael Pradel. *USENIX Security Symposium*

WWW'19 **Anything to Hide? Studying Minified and Obfuscated Code in the Web**
Philippe Skolka, Cristian-Alexandru Staicu, Michael Pradel. *The Web Conference (WWW)*

ICSE'19 **NL2Type: Inferring JavaScript Function Types from Natural Language Information**
Rabee Sohail Malik, Jibesh Patra, Michael Pradel. *International Conference on Software Engineering*

ASPLOS'19 **Wasabi: A Framework for Dynamically Analyzing WebAssembly**
Daniel Lehmann, Michael Pradel. *International Conference on Architectural Support for Programming Languages and Operating Systems*

CACM'19 **Automated Program Repair**
Claire Le Goues, Michael Pradel, Abhik Roychoudhury. *Communications of the ACM*, 62(12), pages 56–65

CSUR'19 **A Survey of Compiler Testing**
Junjie Chen, Jibesh Patra, Michael Pradel, Yingfei Xiong, Hongyu Zhang, Dan Hao, Lu Zhang. *ACM Computing Surveys*, 53(1), pages 1–36

OOPSLA'18 **DeepBugs: A Learning Approach to Name-based Bug Detection**
Michael Pradel, Koushik Sen. *Conference on Object-Oriented Programming, Systems, Languages, and Applications*

OOPSLA'18 **Test Generation for Higher-Order Functions in Dynamic Languages**
Marija Selakovic, Michael Pradel, Rezwana Karim Nawrin, Frank Tip. *Conference on Object-Oriented Programming, Systems, Languages, and Applications*

ASE'18 **How Many of All Bugs Do We Find? A Study of Static Bug Detectors**
Andrew Habib, Michael Pradel. *International Conference on Automated Software Engineering*

ASE'18 **Is This Class Thread-Safe? Inferring Documentation using Graph-based Learning**
Andrew Habib, Michael Pradel. *International Conference on Automated Software Engineering*

FSE'18 **Feedback-Directed Differential Testing of Interactive Debuggers**
Daniel Lehmann, Michael Pradel. *European Software Engineering Conference and Symposium on the Foundations of Software Engineering*

ICSME'18 **Change-aware Dynamic Program Analysis for JavaScript**
Dileep R. K. Murthy, Michael Pradel. *International Conference on Software Maintenance and Evolution*

USENIX Security'18 **Freezing the Web: A Study of ReDoS Vulnerabilities in JavaScript-based Web Servers**
Cristian-Alexandru Staicu, Michael Pradel. *USENIX Security Symposium*

ICSE'18 **ConflictJS: Finding and Understanding Conflicts Between JavaScript Libraries**
Jibesh Patra, Pooja N. Dixit, Michael Pradel. *International Conference on Software Engineering*

NDSS'18 **Synode: Understanding and Automatically Preventing Injection Attacks on Node.js**
Cristian-Alexandru Staicu, Michael Pradel, Ben Livshits. *Network and Distributed System Security Symposium*

CGO'18 **Synthesizing Programs that Expose Performance Bottlenecks**
Luca Della Toffola, Michael Pradel, Thomas R. Gross. *International Symposium on Code Generation and Optimization*, pages 314–326

ASE'17 **Automatically Reducing Tree-Structured Test Inputs**
Satia Herfert, Jibesh Patra, Michael Pradel. *International Conference on Automated Software Engineering*, pages 861–871

ASE'17 **Saying "hi!" Is Not Enough: Mining Inputs for Effective Test Generation**
Luca Della Toffola, Cristian-Alexandru Staicu, Michael Pradel. *International Conference on Automated Software Engineering*, pages 44–49

OOPSLA'17 **Detecting Argument Selection Defects**
Andrew Rice, Edward Aftandilian, Ciera Jaspan, Emily Johnston, Michael Pradel, Yulissa Arroyo-Paredes. *Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 104:1–104:22

PLDI'17 **Systematic Black-Box Analysis of Collaborative Web Applications**
Marina Billes, Anders Møller, Michael Pradel. *Conference on Programming Language Design and Implementation*, pages 171–184

ISSTA'17 **An Actionable Performance Profiler for Optimizing the Order of Evaluations**
Marija Selakovic, Thomas Glaser, Michael Pradel. *International Symposium on Software Testing and Analysis*, pages 170–180

ICSE'17 **Making Malory Behave Maliciously: Targeted Fuzzing of Android Execution Environments**
Siegfried Rasthofer, Steven Arzt, Stefan Triller, Michael Pradel. *International Conference on Software Engineering*, pages 300-311

ICSE'17 **Efficient Detection of Thread Safety Violations via Coverage-Guided Generation of Concurrent Tests**
Ankit Choudhary, Shan Lu, Michael Pradel. *International Conference on Software Engineering*

CSUR'17 **A Survey of Dynamic Analysis and Test Generation for JavaScript**
Esben Andreasen, Liang Gong, Anders Møller, Michael Pradel, Marija Selakovic, Koushik Sen, Cristian-Alexandru Staicu. *ACM Computing Surveys*, 50(5), pages 1–36

EMSE'17 **Pinpointing and Repairing Performance Bottlenecks in Concurrent Programs**
Tingting Yu, Michael Pradel. *Empirical Software Engineering (EMSE)*, 23(5), pages 3034–3071

ICSE'16 **Performance Issues and Optimizations in JavaScript: An Empirical Study**
Marija Selakovic, Michael Pradel. *International Conference on Software Engineering*, pages 61–72

ICSE'16 **Nomen Est Omen: Exploring and Exploiting Similarities between Argument and Parameter Names**
Hui Liu, Qiurong Liu, Cristian-Alexandru Staicu, Michael Pradel, Yue Luo. *International Conference on Software Engineering*, pages 1063–1073

ISSTA'16 **Monkey See, Monkey Do: Effective Generation of GUI Tests with Inferred Macro Events**
Markus Ermuth, Michael Pradel. *International Symposium on Software Testing and Analysis*, pages 82–93

ISSTA'16 **SyncProf: Detecting, Localizing, and Optimizing Synchronization Bottlenecks**
Tingting Yu, Michael Pradel. *International Symposium on Software Testing and Analysis*

OOPSLA'15 **Performance Problems You Can Fix: A Dynamic Analysis of Memoization Opportunities**
Luca Della Toffola, Michael Pradel, Thomas R. Gross. *Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 607–622

FSE'15 **JITProf: Pinpointing JIT-Unfriendly JavaScript Code**
Liang Gong, Michael Pradel, Koushik Sen. *European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 357–368

ISSTA'15 **DLint: Dynamically Checking Bad Coding Practices in JavaScript**
Liang Gong, Michael Pradel, Manu Sridharan, Koushik Sen. *International Symposium on Software Testing and Analysis*, pages 94–105

ECOOP'15 **The Good, the Bad, and the Ugly: An Empirical Study of Implicit Type Conversions in JavaScript**
Michael Pradel, Koushik Sen. *European Conference on Object-Oriented Programming*

ICSE'15 **TypeDevil: Dynamic Type Inconsistency Analysis for JavaScript**
Michael Pradel, Parker Schuh, Koushik Sen. *International Conference on Software Engineering*, pages 314–324