

Alex Potanin

Associate Professor, [School of Computing](#), Australian National University
Adjunct Professor, [S3D](#), CMU · Senior Visiting Fellow, [Trustworthy Systems](#), UNSW Sydney
<https://potanin.github.io/> alex.potanin@anu.edu.au
<https://www.linkedin.com/in/alex-potanin-901686367/>

Summary

I develop programming languages and verification tools for secure software, with 20 years of research in ownership, capabilities, effects and language implementation. I founded and lead the Fiducia¹ project, including hands-on development of its Rust-based tool chain for seL4 systems, and co-created Wyvern² while on my sabbatical at the Carnegie Mellon University. My current research program spans Rust verification, reactive systems and mechanised language semantics, including recent supervision of Loom framework based Lean formalisation of Pancake. I am interested in building reusable verification infrastructure that connects precise specifications to independently checkable proofs.

My current research group works on exactly the problem of proving ordinary code correct: partial borrows in Rust formalised in Aeneas, lock-free and concurrent data structures verified in Iris and Verus, reactive systems proved in F*, quantum programs verified in Dafny, and the Pancake systems language mechanised in Lean. I have run large organisations as well as research projects: I turned the ACM SIGPLAN budget from a \$400K annual loss into a surplus as its Treasurer, and I built and shipped the monitoring systems that two universities use to track thousands of students.

At a Glance

- **Core expertise:** programming language design and implementation, type systems (ownership, immutability, effects, capabilities), software verification, secure systems on seL4, Rust safety and verification
- **Proof assistants and verifiers I used so far:** Lean, Rocq (including Iris), Dafny, Verus, Aeneas, HOL4 (via seL4 and Pancake collaborators)
- **Languages and systems:** Rust (compiler and tooling), C, Java and the JVM, Scala, seL4 Microkit, AArch64 SoC
- **Industry impact:** Rust lifetimes (similar to my PhD on Generic Ownership), Scala capture calculus and CUE modules (collaborated to pass on the design decision we made for Wyvern), security flaws found in JVM servers and CHERI architecture
- **Track record:** 100+ peer-reviewed publications, 6 distinguished or best paper/artifact awards, 80+ students (from RA's to PhD's) supervised, A\$2.8M+ in research funding, Senior Member of the ACM, Permanent Member of IFIP WG 2.4
- **Education:** BSc (Mathematics), BSc Hons, PhD from Victoria University of Wellington, NZ

The Fiducia Project (2024 – present)

Fiducia closes the trust gap between the formally verified seL4 kernel and the applications built on top of it. The key idea is to describe a whole system once, as a process calculus-inspired language of seL4 protection domains, and let the compiler project it into correct local programs while checking information flow and deadlock freedom at compile time. I created the project, lead it, and am involved in the engineering. The team is a Research Fellow ([Dr Yan Liu](#), formerly a formal verification engineer at Runtime Verification and Pi Squared), several PhD and Honours students or research assistants, and partners at Trustworthy Systems (UNSW) and industry.

- **Language design.** A system declaration language (`.fis`) for protection domains, memory regions and typed one-way channels, and a per-domain process language (`.fi`) in the CSP style with events, guarded choice and send/receive. A two-level (Low/High) security type system rejects information

¹<https://fiducia-lang.github.io>

²<https://wyvernleng.github.io>

leaks statically; five well-typedness conditions guarantee that every well-typed system is deadlock-free. The formal semantics (CoreFi) is written up for publication.

- **Compiler and tooling.** A compiler written in Rust that parses, validates cross-references and memory layout, checks information flow, and emits a seL4 Microkit System Description directly. Rich diagnostics, an LSP server and VS Code support; the full per-file pipeline runs in under 4 microseconds. Versioned internal releases through 2.15; public release in preparation. AI tooling was used in completing this work.
- **Verified security patterns.** A growing library (nine so far) of cross-domain guards implemented on seL4 and being moved into Fiducia: network data diode, clipboard guard, multi-level-security email guard, NRL pump against timing covert channels, Linux-VM-to-Linux-VM data diode, in-line JPEG metadata filter, keyboard switch, trusted framebuffer compositor, domain-switch flush. The ultimate goal is to fully verify the seL4 **echo server** example that is used for performance benchmarks in the seL4 Device Driver Framework (SDDF).
- **Hardware bring-up.** Ported seL4, Microkit and libvmm to the Kria K26I, nVidia Jetson Orin Nano and AGX Orin, as well as nVidia Jetson Thor (ARMv9). Wrote the public walkthroughs that others now follow.
- **Robotics demonstration.** Booted seL4 on a Unitree D1 robot arm with the vendor’s unmodified Ubuntu as a guest; the motors are now reachable only through a small geometry-aware filter outside Linux, so root inside the vendor image cannot drive the arm into the desk.
- **Recognition and funding.** Innovation Award Finalist (2nd place, shared) at the ASCA Pitch Day 2026 on Resilient Command and Control, for a *Provably Secure Tactical Edge Command and Control Platform* with Gernot Heiser (UNSW). Department of Defence grant (2027) for *Automated, Compositional and AI-Assisted Verification of Embedded and Cyber-Physical Systems with Fiducia*.
- **AI-assisted verification.** Hands-on experiments using frontier and local open weights (with RAG) LLMs for mechanised proof and for systems porting.

The Wyvern Language (2013 – 2023)

Wyvern is a general-purpose language that treats security and usability as primary design goals, built with Jonathan Aldrich’s group at CMU after my year-long sabbatical there in 2013. The design rests on two ideas that have since travelled well beyond the project: object capabilities as the only way to obtain authority, and effects tracked in the type system so that a module’s permissions are visible in its signature.

- **What I contributed.** Co-designed the language and its type system; led the work on the capability-based module system for authority control (ECOOP 2017), on combining capabilities with effects (*Capabilities: Effects for Free*, ICFEM 2018; *Bounded Abstract Effects*, TOPLAS 2022), and on decidable subtyping for path-dependent types and type members (POPL 2020). Co-authored the type-specific languages work that lets safe DSLs be embedded without injection risks (ECOOP 2014, Distinguished Paper).
- **Industry uptake.** The [CUE](#) configuration language, used widely inside Alibaba’s cloud, based its module system on Wyvern’s. [Scala](#) drew on our path-dependent types and type members and later adopted the capabilities-and-effects combination in its capture calculus, whose ideas originate in our ICFEM 2018 paper.
- **People.** Supervised two of the PhDs that built Wyvern: Darya Melicher (CMU, now Product Manager, previously Google) and Julian Mackay (VUW, now Proof Engineer at Kry10). Twelve-plus co-authored papers; open source at <https://wyvernlang.github.io/>.

Verification and Verified Rust (current work)

- **Partial borrows for Rust.** View types that let the borrow checker accept more provably safe programs, formalised in Aeneas with mechanised proofs (Sasha Pak’s PhD, with collaborators from the National University of Singapore).
- **Concurrent and lock-free data structures in Rust**, verified in Iris (Abolfazl Sharifi’s PhD) and in Verus (Billy Jaffray’s Honours project, 2026). Verified Extract Method refactoring for Rust (Matt Britton’s Honours project, 2025); as well as early automated Rust refactoring tooling

(Garming Sam’s Honours project, 2015).

- **Reactive systems in F***. Pipit, a verified Lustre-style language for proving pre- and post-conditions of reactive controllers (ECOOP 2024, **Distinguished Artifact**), with Amos Robinson, now at Microsoft.
- **Lean and Pancake**. Mechanising the seL4 ecosystem’s Pancake systems language in Lean (Jack Hackshaw’s Honours 2026, continuing as a PhD), and Pancake-by-construction with real driver examples (Jakob Jerebica’s PhD at KIT with Ina Schaefer and Michael Norrish).
- **Correctness by construction**. Traits as a correctness-by-construction mechanism (FORTE 2022, **Best Paper**); immutability and encapsulation for sound information flow control (TOPLAS 2022); flexible CbC (LMCS 2023), all with Ina Schaefer’s group.
- **Quantum program verification in Dafny**. First embedding of quantum program verification into Dafny (OOPSLA 2025) and validation of state preparation programs (ESOP 2026, **Distinguished Paper**); co-PI on a US\$850K NSF GOALI grant (2026–2030) with Liyi Li, Iowa State.
- **Earlier mechanisation**. Featherweight Java with assignment and immutability encoded in Coq (FTfJP 2012); bytecode verification of `@NonNull` types for the JVM (CC 2008, Science of Computer Programming 2011).

Ownership, Immutability and the Road to Rust

My PhD showed that type polymorphism alone is enough to give a language ownership types: *Generic Ownership for Generic Java* (OOPSLA 2006, 100+ citations) is the design Rust later adopted as lifetime parameters. With Michael Ernst’s group at MIT I extended the same trick to immutability (*Object and Reference Immutability using Java Generics*, FSE 2007, **ACM SIGSOFT Distinguished Paper**, 100+ citations) and then to both at once (OOPSLA 2010). The Marsden-funded project I led in 2008–2012 unified ownership and immutability in one framework, summarised in the *Immutability* chapter of Springer’s *Aliasing in Object-Oriented Programming* (2013). An ICSE 2013 study measured the real cost of encapsulation in practice, which is the question every ownership system has to answer for engineers.

Security Research with Industry Reach

- **EvilPickles** (ECOOP 2017, **Distinguished Artifact**): denial-of-service attacks by object-graph engineering, which uncovered flaws in Java-based servers (Jenkins, Tomcat, JBoss) and their .NET equivalents; funded by Oracle Labs (US\$50,000).
- **CHERI capability hardware**: a security analysis of CheriBSD and Morello Linux (ESORICS 2026), plus student projects on CHERI exploits and obfuscation.
- **Capabilities for runtime invariants** (Science of Computer Programming 2022) and type-specific languages against injection attacks (HotSoS 2014).

Employment

2022 – Present	Associate Professor, Australian National University; convenor of the BE Software Engineering; Associate Director HDR (2023–2025) for 200+ PhD students
2026 – Present	Senior Visiting Fellow, Trustworthy Systems, UNSW Sydney
2023 – Present	Adjunct Professor, S3D, Carnegie Mellon University, USA
2026	Research Consultant, MetaCognition AI, Canberra
2025	Research Consultant, Skykraft, Canberra (trustworthy systems for space platforms)
2025	Research Consultant, Usability Dynamics Inc, NC, USA
2006 – 2022	Lecturer, Senior Lecturer, then Associate Professor, Victoria University of Wellington; Deputy Head of School (2022, 15 direct reports); Associate Dean (Students) (2021–2022); Program Director for Software Engineering (2017–2018)
2019/20	Visiting Associate Professor, Kyoto University, Japan (sabbatical)
2013	Research Scholar, Carnegie Mellon University (sabbatical; start of Wyvern)
2005 – 2006	Software Engineer, Innaworks Limited, Wellington (start-up)
2004	Visiting Researcher, Purdue University, USA
2001 – 2003	Software Engineer (part-time), Catalyst Systems Limited, Wellington

Leadership and Operations

- **Treasurer and Member at Large, ACM SIGPLAN Executive Committee (2024–2027).** Turned the budget around from a roughly \$400K/year loss to a sustainable \$100K–\$200K/year surplus by synchronising conference registrations and expenses, writing clear policies, and coordinating across steering committees.
- **Conference leadership.** Chair of the SPLASH Steering Committee (2024–2026); General Chair of SPLASH/OOPSLA 2022; Research Committee Co-Chair of OOPSLA 2024; Program Committee Chair of APLAS 2025; General Chair of APLAS 2018. Regular programme committee member for OOPSLA, PLDI, ECOOP, ESOP and APLAS.
- **Systems I built and shipped.** An HDR monitoring app now used by ANU School of Computing staff to track the progress and wellbeing of 200+ PhD students; a Student Engagement System at VUW that flagged struggling students early across a faculty of about 2000.
- **Managing people.** Fifteen academic direct reports as Deputy Head of School; led the review and re-accreditation of VUW’s largest engineering major with 20+ academics; led the introduction of VUW’s postgraduate AI and ML program.

Selected Publications

1. Liyi Li, Anshu Sharma, Zoukarneini Difaizi Tagba, Sean Frett, Alex Potanin. *Validating Quantum State Preparation Programs*. ESOP 2026. **Distinguished Paper.**
2. Dariy Guzairov, Alex Potanin, Stephen Kell, Alwen Tiu. *A Security Analysis of CheriBSD and Morello Linux*. ESORICS 2026.
3. Feifei Cheng, Sushen Vangeepuram, Henry Allard, Seyed M.R. Jafari, Alex Potanin, Liyi Li. *Embedding Quantum Program Verification into Dafny*. OOPSLA 2025.
4. Sasha Pak, Fabian Mühlböck, Alex Potanin. *A Verified Thread-Safe Array in Rust*. IWACO 2025.
5. Amos Robinson, Alex Potanin. *Pipit on the Post: Proving Pre- and Post-Conditions of Reactive Systems*. ECOOP 2024. **Distinguished Artifact.**
6. Tobias Runge, Alex Potanin, Thomas Thüm, Ina Schaefer. *Traits: Correctness-by-Construction for Free*. FORTE 2022. **Best Paper.**
7. Tobias Runge, Marco Servetto, Alex Potanin, Ina Schaefer. *Immutability and Encapsulation for Sound OO Information Flow Control*. ACM TOPLAS 45(1), 2022.
8. Darya Melicher, Anlun Xu, Valerie Zhao, Alex Potanin, Jonathan Aldrich. *Bounded Abstract Effects*. ACM TOPLAS 44(1), 2022.
9. Julian Mackay, Alex Potanin, Jonathan Aldrich, Lindsay Groves. *Decidable Subtyping for Path Dependent Types*. POPL 2020.
10. Aaron Craig, Alex Potanin, Lindsay Groves, Jonathan Aldrich. *Capabilities: Effects for Free*. ICFEM 2018.
11. Darya Melicher, Yangqingwei Shi, Alex Potanin, Jonathan Aldrich. *A Capability-Based Module System for Authority Control*. ECOOP 2017.
12. Jens Dietrich, Kamil Jezek, Shawn Rasheed, Amjed Tahir, Alex Potanin. *EvilPickles: DoS Attacks Based on Object-Graph Engineering*. ECOOP 2017. **Distinguished Artifact.**
13. Garming Sam, Nicholas Cameron, Alex Potanin. *Automated Refactoring of Rust Programs*. ACSC 2017.
14. Cyrus Omar, Darya Kurilova, Ligia Nistor, Benjamin Chung, Alex Potanin, Jonathan Aldrich. *Safely Composable Type-Specific Languages*. ECOOP 2014. **Distinguished Paper.**
15. Alex Potanin, Johan Östlund, Yoav Zibin, Michael D. Ernst. *Immutability*. In *Aliasing in Object-Oriented Programming*, LNCS 7850, Springer, 2013.
16. Yoav Zibin, Alex Potanin, Mahmood Ali, Shay Artzi, Adam Kiezun, Michael D. Ernst. *Object and Reference Immutability using Java Generics*. FSE 2007. **ACM SIGSOFT Distinguished Paper.**
17. Alex Potanin, James Noble, Dave Clarke, Robert Biddle. *Generic Ownership for Generic Java*. OOPSLA 2006.

Full list of 100+ publications: <https://potanin.github.io/publications.html>.

Awards

- ESOP/ETAPS 2026 Distinguished Paper; ECOOP 2024 Distinguished Artifact; FORTE 2022 Best Paper; ECOOP 2017 Distinguished Artifact; ECOOP 2014 Distinguished Paper; ESEC/FSE 2007 ACM SIGSOFT Distinguished Paper
- 2026 – Innovation Award Finalist (2nd place, shared), ASCA Pitch Day 2026, Resilient Command and Control
- 2005 and 2003 – 2nd Prize and Judges’ Prize, ICFP Programming Contest; 2003 – 2nd Place, ACM International Student Research Competition
- 1997 – Finalist, George Soros International Mathematics Olympiad; 1996 – Winner, Russian City Tournament in Mathematics

Students and Alumni

I have supervised more than 80 students, currently a postdoc, twelve PhD students and an Honours student across ANU, KIT and Iowa State. Former students and postdocs are now at Microsoft (Amos Robinson), Kry10 (Julian Mackay, proof engineer), Google and then product management (Darya Melicher), Xero (Felix Shi, senior security architect), Westpac (Garming Sam, security platform engineer), and in faculty positions at Nottingham Malaysia (Radu Muschevici) and the Czech Technical University (Paley Li).

Teaching

Over twenty years I have taught compiler engineering, algorithms, software engineering and mobile development to classes from twenty to five hundred, with consistently strong evaluations, and I have designed or rebuilt seven courses. At ANU I turned the 400-student capstone (TechLauncher) into a time-boxed, in-person model that halved the tutor budget while raising student and client satisfaction, and I rebuilt the second-year software engineering course so that every one of 50+ teams lands a pull request in an established open-source project. I lead the School of Computing’s GenAI teaching deployment under the ANU–Anthropic partnership (US\$355K in API credits, 2026–2027).

Referees

Professor Peter Müller (ETH Zurich), Professor Steve Blackburn (ANU and Google DeepMind), Professor Jeff Foster (Tufts), Professor David Lo (SMU), Professor Andrew Myers (Cornell), Professor Anders Møller (Aarhus), Professor Sophia Drossopoulou (Imperial), Professor Jan Vitek (Epic Games). Contact details on request.