Podcast Name: *ACM ByteCast* Episode: Xavier Leroy - Episode 57

Welcome to the *ACM ByteCast* podcast, a series from the Association for Computing Machinery! The podcast features conversations with researchers, practitioners, and innovators at the intersection of computing research and practice about their experiences, lessons learned, and visions for the future of computing. In this episode, we are joined by Xavier Leroy, a distinguished French computer scientist and a prominent developer of the OCaml programming language. He is currently a professor at the Collège de France and formerly worked at INRIA, a leading French research institute in computer science. Among his numerous accolades are the 2015 ACM Fellowship and awards from the Royal Society and ACM for his contributions to software systems and programming languages.

Kicking off the conversation, Xavier shares about the evolution of Ocaml, a system of which he was a key developer. He recounts his early experiences with ML-like languages during his time at INRIA, where he was introduced to the Caml language, an early ML variant. Leroy describes how he and a fellow student created a simplified, lightweight implementation of Caml, which was both a learning experience and the first open-source version of the language. This version was designed to be practical, running on PC's and Mac's with minimal resources, and served as an effective tool for teaching and research. As OCaml evolved, Leroy and his colleagues aimed to create a high-performance compiler with features like a new module system and support for object-oriented programming, while maintaining a lightweight and efficient runtime system. OCaml became suitable for a wide range of applications, from symbolic processing to distributed systems, web development, real-time trading, and blockchains. The system was adopted by Jane Street Capital for its financial applications and others for industrial applications.

Then, the conversation shifts to formal verification, a field that has historically faced skepticism from industry due to its perceived complexity and cost. Leroy acknowledges that while formal verification has made slow progress since its inception in the 1960s, it remains crucial for high-risk applications, particularly in life-critical systems. He also discusses the use of formal methods in cloud infrastructure at companies like Amazon AWS and in projects like his own CompCert, a formally verified C compiler. However, formal verification is not yet widespread due to the high costs and the need for mathematical specifications, which are often lacking in many application areas.

There is a growing belief that programming as we know it might change significantly or even disappear because of the capabilities of AI tools like Copilot. The discussion then delves deeper into the limitations of AI-generated code, emphasizing that these tools are highly dependent on the training data available. AI systems often excel at generating code in well-documented and widely used languages but falter with more specialized languages. Xavier expresses concern that while AI can automate the enjoyable aspects of coding, such as writing small functions, it falls short in more critical areas like system architecture, code review, and validation. He suggests that AI could be more useful in generating test suites or performing code reviews, tasks that require a different kind of expertise than writing code.

Next, the conversation focuses on ACM's efforts to become a more global and diverse organization. While ACM has made strides in involving European academics in programming languages research, there is still work to be done in connecting with researchers in Asia, particularly in Japan and South Korea. More initiatives, such as programming competitions and mentoring sessions, could help expand ACM's reach and influence in these regions and to younger demographics.

The final segment of the conversation addresses the gap between academia and industry, particularly in the field of formal methods and software development. Xavier emphasizes the importance of personal contacts and networking in fostering collaboration between academics and industry professionals. The discussion concludes with a call for more opportunities for networking and collaboration, acknowledging that while some efforts are being made, there is still a significant gap to bridge.

Key takeaways:

- 1:53 The evolution of the Ocaml system.
- 7:17 Ocaml in industry and financial applications.
- 9:39 Formal verification and its role in industry.
- 16:41 Discussing AI and software development.
- 17:52 The limitations of AI in code generation.
- 24:53 Expanding ACM's global and demographic reach.
- 27:16 Bridging the gap between academia and industry.

Links

Learn more about <u>Xavier Leroy</u>. Learn more about the ACM ByteCast podcast at <u>https://learning.acm.org/bytecast</u>