The Future of Software Engineering Beyond the Hype of AI

Steven Fraser, Innoxec, Santa Clara, CA, USA, sdfraser@acm.org
Dennis Mancl, MSWX Software Experts, Bridgewater, NJ, USA, dmancl@acm.org

ABSTRACT

ACM/IEEE ICSE 2025, the world's premiere software engineering conference, celebrated its 50th year. To mark the occasion, a panel of academics and practitioners was convened to discuss the future of software engineering. The panelists shared their views of how software engineering jobs will change, how software will continue to have an impact on society, and whether AI could be trusted to create secure software. The panel was organized and moderated by Steven Fraser (Innoxec) and Dennis Mancl (MSWX Software Experts). Panelists included Adriana Meza Soria (MIT-IBM Watson AI Lab), Landon Noll (independent), Tom DeMarco (The Atlantic Systems Guild), Laurie Williams (North Carolina State), Mary Shaw (Carnegie Mellon), Irene Manotas (IBM), and Bertrand Meyer (Eiffel Software).

INTRODUCTION

In In 2025 the International Conference on Software Engineering (ICSE) celebrated 50 years since its first edition in Washington DC. A celebratory panel at Ottawa's ICSE 2025 was organized by Steven Fraser and Dennis Mancl to discuss the future of software engineering in the context of the hype of AI. Discussions focused on the promise and threats of automation on software development and society in general.

AI was much in evidence on the ICSE 2025 conference program. AI-based development tools and applications play an increasing role in today's world, raising issues of trust, security, and resilience. One question on many attendees' minds: What will be the role of software engineers in the future?

PANELIST POSITIONS

Panelists began by sharing their vision for the future of software engineering. They elaborated on position statements published prior to the conference [1]. Panelists noted both the value of generative AI tools and the risks of trusting automated code generation. One notable concern expressed was whether statistically based AI could adequately address security and privacy issues with generated software.

Adriana Meza Soria (MIT-IBM Watson AI Lab) is part of the Granite team working on Large Language Models for code intelligence. Adriana believed that AI tools will become an essential tool for human developers. "It will be less about replacing developers and more about reimagining their development experience."

Landon Noll is an independent consultant and an expert in software security. He explained that the software community is going through a "hype cycle" with respect to AI, similar to past emergent technologies. He further observed that many industry software developers lack the professional discipline necessary to design and implement resilient systems.

Tom DeMarco (The Atlantic Systems Guild) is a software consultant and co-author of *Peopleware*. He believed that today's software developers will still have jobs, "because there is a lot of low hanging fruit as we put AI into various parts of businesses." However, he was concerned for the next generation of software developers. Tom hoped that we will introduce AI without killing the job market. He assumed that the need for new skills will require companies and governments to make investments.

Laurie Williams (North Carolina State University) is a software security researcher. Laurie warned that AI might not create quality code. "With LLMs and AI, we can get more efficient in producing code, but not necessarily code that does what we want it to do, and/or secure code."

Mary Shaw (Carnegie Mellon University) believed that AI tools are another step in raising the level of abstraction in software design. She identified fundamental limitations in what AI can do. "Design and engineering knowledge has an enormous amount of tacit information. We have abstractions, we have principles that are not well documented, and much of it is just tacit."

Irene Manotas (IBM's AI for Code group) explained that we need to look beyond code: "We need to empower software engineers for certain tasks: not just for coding, but for designing, maintenance, and testing."

Bertrand Meyer (Eiffel Software) suggested tasks where AI-assisted development is useful. "The absolutely ideal case is you are trying to build an application with an API you don't know." He gave the example of writing a graphical application using an open-source graphics library. Bertrand explained that rather than sifting through the API documentation, a developer can get a first-draft application by asking a few questions to a chatbot. But he warned that there will be no guarantee of correctness.

IS AI SOFTWARE TRUSTWORTHY?

Throughout ICSE, conference attendees expressed concerns regarding the challenge of software development with AI-based code generation. In his opening ICSE 2025 keynote, David Parnas spoke on the "Regulation of AI and Other Untrustworthy Software." Parnas was critical of the quality of all AI-based products [2] and he believed that AI is detrimental to product safety. "AI is not afraid of going to jail," was a comment shared by one academic at the keynote.

The panel session audience posed several probing questions: Is software engineering still important? How will companies change their recruiting strategies for software staff? How should startup companies balance high-tech tools and software engineering principles? How can we ensure that our software engineering work has a positive impact on society? Are the claims of high productivity for AI-based development real?

THE FUTURE OF SOFTWARE

Both Mary and Landon gave strong arguments for evolving software engineering into an engineering discipline. Software development should not be a race to have undisciplined developers or automated tools deliver millions of lines of code. Landon worried that startup companies have embraced a culture of "fail fast and break things and go forward." Landon said that this culture impairs the ability of software to be correct. We face four challenges: the customer isn't being exact in what they want, engineers don't hear what the customer is saying,

the specifications aren't exact, and testing is a last-minute thing that gets squeezed out of the schedule.

Mary responded that "programming from a formal base" isn't always possible. We would like to get a solid specification, then write programs in a sound language, and finally prove that it's correct – and then we're good to go. "The problem is the world doesn't really quite work that way most of the time."

She observed that we have a huge number of "vernacular programmers" today – these are "people who are developing [software] for their own needs, not because they are professional programmers." They outnumber highly-trained professional software developers by at least an order of magnitude. The software they write is often exploratory and they are not writing critical applications where the consequences of failure are catastrophic. Vernacular programmers code "with the purpose of understanding what the software can do for them and what they really need."

Mary believed that we should accept both ways of building software: "programming from a specification toward code" and "programming toward a specification."

THE AI HYPE CYCLE

Landon explained that when new tools and techniques emerge, there are two reactions. Some experts in technology and business will advocate a "hype thesis" – they proclaim that the new thing is "the next great technology." Others line up with an "anti-thesis" – "pointing out the flaws, implementation problems, and the failure of practitioners to understand the context that sparked the hype cycle in the first place."

Landon thought that hype cycles are inevitable in the world of science and technology. A successful tool or process will lead to compromise: then both the thesis and the anti-thesis "merge" into a working synthesis. The synthesis will endure until it is overshadowed by a new hype cycle. Landon claimed that "hype cycles are not inherently bad, although it is hard to tell when one is at the peak of the thesis while those calling out the anti-thesis are ignored by the populus."

Mary thought that AI will push us to develop new ways to understand the "correctness" of software. She explained that much of the criticism of generative AI is about its potential unreliability: the models are both opaque and non-deterministic. Mary pointed out that this isn't a fatal flaw since we have applied engineering approaches to deal with reliability and non-determinism in other contexts.

THE FUTURE OF SOFTWARE SKILLS

Irene suggested that AI hasn't changed development skills. "Job requirements won't change much, but we will need knowledge about AI." Irene claimed that AI will improve productivity, but it will also require software engineers to use different strategies. We must also learn about the limitations of the new tools. Laurie added that the increased use of "AI for Software Engineering" will require developers to do more critical thinking; a developer "needs to be knowledgeable and skeptical enough to know that what comes out of an AI model is right."

Adriana envisioned a future where developers collaborate closely with AI-based tools. “It will be like having a powerful collaborator.” That future may be already taking shape – as developers are increasing their use of AI code assistants, many of them available in mainstream software tools. Adriana also anticipated that generative AI will drive innovation in no-code development applications, empowering vernacular programmers to independently co-create high-level prototypes.

Bertrand noted that we still struggle with requirements: we don’t always understand the requirements of the systems we are developing. He warned that when we use AI, we might become even further removed from the problem domain. Writing requirements is at least as hard as writing programs. So although we appear to be progressing in code generation, we aren’t really solving the overall problem of system development.

Adriana agreed, noting that developing prototypes and MVP-level systems differs significantly from building production-ready systems. She emphasized that building systems that incorporate LLMs and AI agents will require more orchestration than building traditional software. Developers’ knowledge of software architecture, design, security – combined with a deeper understanding of generative AI capabilities and limitations – will be more critical than ever to maintain software quality. Adriana also shared that AI-based software systems will likely involve not just one but multiple LLMs, each contributing to different objectives. For instance, some models may focus on fulfilling functional requirements, while other models may address safety, ethics, and usability concerns.

SOCIETAL IMPACT

Software has evolved, and changes in society have increasingly focused on the user experience, diversity, and privacy. An audience member inquired – how should development practices be connected to societal issues?

Tom thinks that AI is helping to drive a profound change in technology. We may not be able to get perfect answers to everything, but we should accept technologies that make things better. For example, self-driving cars guided by AI might be safer than cars with human drivers. Tom believes in finding “heuristic approaches” for societal problems.

Mary agreed with Tom, citing “the Wicked Problems paper” by Horst Rittel and Melvin Webber [3]. There are many complex problems where it is better to work progressively towards a solution and consensus. Whenever there are multiple stakeholders who articulate and require conflicting needs, a standard engineering process is doomed to never complete. However, it’s important to understand the consequences of failure and to limit trust accordingly. If failure could cause great damage, human oversight should be required.

Bertrand wanted people to tell the truth instead of just accepting the hype. He referenced an ICSE 2024 paper about code completion models by several researchers from Delft University of Technology [4]. Code completion should save developers time and effort in theory, but this paper reported that developers had a low “acceptance rate” for completions proposed by the tools. The authors also noted that many accepted completions needed to be edited. Today’s code completion tools don’t match the hype.

Laurie believed in vigilance. Technologies should be used for good, but too many good technologies have been used for bad, for example, anonymous networks used as porn distribution sites. The software industry must be responsible; “we need to build in some checks and balances and guard rails, so that the functionality is good for society.”

SOFTWARE PRODUCTIVITY & AI

An audience member commented that he receives frequent email messages praising the productivity gains for AI-based software development. This suggests that AI isn’t hype, it is reality. What should students, faculty, and industry experts study to harness AI technology positively?

Mary reported that her colleagues using AI development tools found a downside for AI-generated code segments. Tool-based suggestions need to be reviewed and modified as necessary. The reports indicate that the impact on productivity may be marginal. Mary suggested that we need better measurements: “We are flush with anecdotes and we are short of data. Before we go charging off in some direction, we’d better make sure to know what we’re counting.”

Landon thought that the next “hype” beyond AI will be agents that provide assistance to the development process. However, he is concerned that even if compilers, development environments, and testing harnesses can call on an agent to provide the needed context, it may not be enough. Humans must still be in the loop to make engineering judgments about what development choices are acceptable and what might introduce an unacceptable risk of harm.

Bertrand reminded us about the cost of the whole software lifecycle. He suggested that solving the coding issue is only part of the problem.

Adriana was concerned about reactions to AI technology in software engineering, ranging from denial (“too many hallucinations”) to panic (“AI will take my job”). She hoped that the software community will move away from negativity. She advocated for a middle ground approach, using AI tools selectively to help individuals be more productive. Adriana shared a guiding principle: “I use AI for tasks that I don’t feel great at.” She turned to AI tools for refining written content to make it more eloquent and engaging, parallelizing sequential code routines, and guiding application modernization work.

Adriana also acknowledged that it’s perfectly normal to feel uneasy about disruptive change. Adriana quoted Pablo Picasso: “The meaning of life is to find your gift. The purpose of life is to give it away.” Adriana explained, “Sometimes my technical side feels it’s giving away a lot, because I spend much less time writing code. But what keeps me moving forward is the big picture.”

SECURE SOFTWARE

Laurie raised another issue: “In the general case, an AI won’t generate secure code. You can do an experiment yourself. Ask your chatbot ‘Write me a program to do X.’ Then when it is finished, ask it ‘Write me a secure program to do X.’ It will be different.”

Landon went one step further: “A sad thing is that you can take a program and ask ‘make me a secure program,’ and then say ‘identify ways to exploit and compromise the integrity of the code’ – and you’ll get more information.” We have to remember to ask that second question.

Irene saw an opportunity for us to make better models. We need to train our models to be more aware of security and vulnerabilities. In addition, we need to develop solutions and strategies to help discover these vulnerabilities in the code so that developers can fix them early.

Laurie warned that we shouldn’t make LLMs the answer for everything. When an ICSE paper reports that an LLM-based tool is used for code inspection and it finds 12 vulnerabilities in a code base, the researchers must ask the question “What other tools are applicable? Should we try using a static analysis tool?” A static analysis tool might find even more problems, and it might be a more reliable piece of an inspection process.

SUMMARY

Tom expressed a belief in “better networking” – we make more progress when people get together and collaborate. When he attended his first ICSE in 1976, Tom was thrilled to have the opportunity to meet some of the stars of software engineering research. At that time, Tom’s peers at Bell Labs were all busy “putting out software all day every day.” But they didn’t network. Many of his colleagues didn’t see much value in talking with academic experts. Tom began to take a different view based on his ICSE experience. He enjoyed having many connections with academics who were willing to share their ideas. Tom admitted that he “started to realize that this is what the academics had going for them – they were better networkers than us.”

Adriana expressed a positive view of the future roles for software engineers. Software engineers will contribute with their rich domain and tacit knowledge about the software engineering field. They will also orchestrate the development of agents, frameworks, protocols, and new tools.

Irene believed that there is an opportunity for our community to keep innovating. Not only for coding, but also for methodologies. Irene asked how do we integrate agents to keep humans in the loop?

Landon thought that we spend too much energy worrying about the current AI hype cycle. He predicted that in ten years the global software development landscape will have progressed through at least one or two new hype cycles. To be prepared for the challenges of these cycles, Landon suggested we will need software practitioners educated in the fundamental principles of computation, the discipline of engineering, and the scientific underpinnings of computer science.

Mary asserted that our greatest challenge is to figure out what we mean when we say things like “correct.” She said that the criteria for a piece of software being acceptable depends on context, including how the software will be used.

Bertrand predicted that new AI tools may help produce software specifications. But he didn’t see how it can help with verification. “We’ll see. I hope to be disproved.”

Laurie warned that cybersecurity is critical. “Attackers are certainly using AI to launch attacks. They [attacks] are up seven times over the last year. So we need as engineers to counter the attacks by writing secure code and using LLMs or any technique possible to get the vulnerabilities out so that the attackers don’t win with their use of AI.”

Takeaways from the panel included:

- Software tools based on AI shouldn’t be trusted to generate correct or secure code.
- Humans with software engineering and AI skills will be needed in the future.
- Be skeptical of reports of productivity gains driven by AI or other tools: reports need to be carefully validated.
- Software engineering requires teamwork and collaboration.
- Humans need to assess and mitigate possible negative impacts of AI on society.

REFERENCES

[1] ICSE 2025 conference. 2025. Panel: The future of software engineering beyond the hype of AI - ICSE 2025 panel 1 position statements. Retrieved May 18, 2025 from https://conf.researchr.org/info/icse-2025/panel:-the-future-of-software-engineering-beyond-the-hype-of-ai

[2] David Parnas. 2025. Regulation of AI and other untrustworthy software. Video. (30 April 2025). Retrieved May 18, 2025 from https://www.youtube.com/watch?v=YyFouLdwxY0

[3] Horst W. J. Rittel and Melvin W. Webber. 1973. “Dilemmas in a General Theory of Planning.” Policy Sciences 4, 155-169. https://doi.org/10.1007/BF014057301973

[4] Maliheh Izadi, Jonathan Katzy, Tim Van Dam, Marc Otten, Razvan Mihai Popescu, and Arie Van Deursen. 2024. Language Models for Code Completion: A Practical Evaluation. In Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (ICSE '24). Association for Computing Machinery, New York, NY, USA, Article 79, 1–13. https://doi.org/10.1145/3597503.3639138

This report is based on the ICSE panel session held on May 1, 2025 in Ottawa, Canada. The panel report appeared in the July 2025 issue of SIGSOFT Software Engineering Notes: https://doi.org/10.1145/3743095.3743102