





# The Future of Software Engineering: Where Will Machine Learning, Agile, and Virtualization Take Us Next?

Dennis Mancl<sup>1</sup>  and Steven D. Fraser<sup>2</sup> 

<sup>1</sup> MSWX Software Experts, Bridgewater, NJ 08807, USA  
dmancl@acm.org

<sup>2</sup> Innexec, Santa Clara, CA, USA

**Abstract.** Software has become the lifeblood of the 21st century, enabling a broad range of commercial, medical, educational, agricultural, and government applications. These applications are designed and deployed through a variety of software best practices. With the onset of the COVID-19 pandemic, developers have embraced virtualization (remote working) and a variety of strategies to manage the complexity of global development on multiple platforms. However, evolving hazards such as network security, algorithm bias, and the combination of careless developers and deliberate attacks continue to be a challenge. An XP2021 panel organized and chaired by Steven Fraser debated the future of software engineering and related topics such education, ethics, and tools. The panel featured Anita Carleton (CMU's SEI), Priya Marsonia (Cognizant), Bertrand Meyer (SIT, Eiffel Software), Landon Noll (Independent Consultant), and Kati Vilkki (Reaktor).

**Keywords:** Agile · AI · Applications · Collaboration · Education · Machine learning · Professionalism · Remote working · Societal needs · Software engineering

## 1 Introduction: The Panelists Share Their Views of the Future

Software development has evolved over the past seventy-five years to meet the changing challenges of software product development. Software development has embraced many innovations in technical and business practices: structured programming, waterfall development processes, OO design, automated testing, outsourcing, open source, networking, offshoring, and Agile methods. New programming languages, SDKs (software development toolkits), and code management tools have improved productivity. Today, there is hope that emerging technologies such as machine learning, virtual communication, and remote working tools will improve the reliability and resilience of software systems.

Technology needs to address a crisis of complexity. Yesterday's batch-oriented mainframe systems were relatively simple, but today's software developers face more complexity. Developers work in a global environment targeting multiple platforms while

managing ubiquitous networks, petabyte datasets, and emergent hazards. New technology could be useful in this complex environment. For example, some of work of developers might be replaced by automated development and machine learning [1, 2]. In this panel session, the panelists expressed a wide spectrum of opinion on the future.

The panelists focused less on technology and more on the ongoing challenges of the software industry. Software teams everywhere must face issues connected to quality, education, outsourcing, ethics, a push to democratize development, and more connections between computing and non-technical fields. The panelists expressed their anticipation for the benefits of cloud technology, ubiquitous computing, smarter development environments, and systems that leverage natural language processing. The panelists expressed concern about two missing pieces in the software industry. First, there is a need for better-trained software professionals who understand the discipline of product development in light of the challenges of security, safety, and ethical economic viability. Second, industry needs an incentive system that not only rewards innovation – but also penalizes companies that knowingly deliver systems that fail or worse yet cause bodily harm.

## 1.1 AI and Machine Learning

Two panelists, Anita Carleton (Software Engineering Institute) and Priya Marsonia (Cognizant), embraced the adoption of new technologies, including artificial intelligence and machine learning, to extend the capacity of software developers. Anita is Director of SEI’s Software Solutions Division, and Priya is a Senior Client Partner at Cognizant.

Anita anticipated that AI will support the development of software systems. She expected that the combination of human and machine intelligence will enable the software industry to keep ahead of a dynamically changing environment. AI-based development tools will help meet increasing quality requirements.

Priya was convinced that we will see more natural language programming and better tool support for developers. She explained, “In the agile world, I see the infrastructure as helping us more,” with humans collaborating with intelligent agents and machine-identified “digital twins.” A future software development environment might be able to analyze the programming styles of team members, then assign pair programming partners based on similarity of programming style. In an agile environment of the future, a developer may pair program with an intelligent agent-selected partner, or even an AI instead of a human partner.

Priya anticipated that another fruitful area for automation is to provide support for large multi-team projects. Systems could scan enormous volumes of complex subsystems, assessing code similarities and offering suggestions of where teams may need to collaborate. Inter-team communication can avoid duplication, discover useful lessons from the past, or improve the testing process. Priya added, “The whole notion of getting suggestions and spotting trends would be embedded in our environment, and we would have to do fewer and fewer rudimentary operations.”

## 1.2 Conventional Technology with Better Failure Analysis

Bertrand Meyer (Schaffhausen Institute of Technology and Eiffel Software), agreed with the positive assessment of the future, but believed that more conventional technology and processes will continue to advance.

Bertrand asserted that there are reasons to be proud of what software has done (including virtual conferences). “Imagine the depth of the stack just to have this [this panel session].” But we have also seen big failures. He referred to the 7-hour outage of emergency services telephone service in France in the first week of June 2021 [3]. “This was a software bug. Why don’t we have the same kind of analysis we have for airplane accidents?” One of software engineering’s biggest challenge is “correctness” – and a public airing of significant bugs will help developers to understand the root causes of future software failures.

## 1.3 Need to Address Software Failures

Landon Noll (Landon Noll and Associates) was more pessimistic. He warned that we are already in a software crisis, and we need major changes to improve the education and management of software professionals to increase future “readiness.”

Landon explained his views – and why he believes that skills and best practices are not improving. “So many companies and developers use the term software engineer – yet [they] lack any formal training or certification to understand even basic engineering principles.” This is a view shared by software engineering thought leaders, such as David Parnas and Mary Shaw [4][5][6]. Landon complained that companies in non-critical industries don’t care, and he blamed the limits on legal liability for software that fails – and users accustomed to accepting poor quality software. Landon claimed that we need to address these skill deficits and fix these economic incentives to make progress.

## 1.4 Less Outsourcing in Data-Driven Industries

Kati Vilkki (Reaktor) shared her consulting experience in digital transformations based on her years as an agile technology leader at Nokia. Kati observed that many banks and insurance companies have gradually “outsourced” many of their information technology functions. At the time, company executives felt that IT was not part of the company’s core business. But today, the pendulum is swinging the other way. These companies have found that their businesses have evolved and that “data is king.” Kati reported: “I see them desperately trying to in-source, to hire software developers.”

## 2 Democratization of the Software Industry?

Priya noted that there is an increased “democratization” of software development, with the rise of many new low-code or no-code environments that can be used by non-programmers to build applications. She talked about Cognizant’s think tank “Center for the Future of Work,” which was launched to study the influences of globalization, virtualization, AI, and the cloud [7]. That future for software engineering, according to

Priya, may be “founded on natural and spoken language, where we don’t have to be as conversant in an arcane coding language, or the vagaries of a very specific environment that’s complex to manipulate.”

Anita added her observations on the evolution of the software industry. She described significant advances in the past 30 years of software development, such as a better focus on architecture, improved practices and tools, agile processes, and “DevSecOps” which have combined to make product development better.

Landon voiced concerns. He noted that, “Agile practices have a potential for helping,” but observed that developers are often attracted by the glitter of new programming languages. He would prefer developers to spend less time on the “language of the month” or “current design fads” and more on engineering principles, best practices, and good algorithm design.

### 3 Can AI and Machine Learning Help?

The panel was split on the question of the potential effectiveness of AI-centric tools.

In 2020, Anita was one of the guest editors of the July/August 2020 issue of *IEEE Computer* [8], which contained several articles that explored future interactions between AI and software engineering. Anita expected that AI will be part of “societal-scale systems.” However, she worried that we still aren’t sure where AI research will take us. Priya believed that future programming infrastructure will incorporate machine learning technology. But Bertrand was skeptical about the potential impact of machine learning on software development. He explained that the development of correct programs requires logic and reasoning, not heuristic search. “Machine learning works on statistical principles. Program correctness works on a different kind of mathematics – which is logic. It’s not so easy to see how we can apply statistical techniques. What does it mean to say the programs in the world are going to be 2% more correct?”.

### 4 Companies May Need to Do More In-Sourcing

Kati worried about the consequences of “out-sourcing followed by in-sourcing.” When companies outsourced most of their development, many of them were seeking to reduce costs, so they collected many applications from different vendors. “I don’t think it’s a wonder that the whole thing is a mess.” The process of in-sourcing and hiring software developers is difficult and costly.

Kati shared her experience in Finland – developers are expensive, they can choose where they want to work, and the companies do not have much experience leading and managing software projects. Some accommodation is needed on all sides. “For this to work,” according to Kati, business specialists need to know more about software and software people need to know more about business. She mentioned the increased demand for her consulting services, where leadership teams frequently request her introductory seminar course on software management.

## 5 Do We Need to Reform the Software Industry?

Three areas of evolutionary or revolutionary change were on the minds of the panelists: reform of the industry's commercial incentives (including product liability), the future of free and open source software, and general improvements in software engineering education.

### 5.1 The Software Industry and Commercial Incentives

Landon believed the failure of modern software engineering is rooted in commercial incentives. He called for reform: a specific list of actions to change the economic model for software development to reduce company incentives to ship software systems and applications that crash. Landon's list:

- Invalidate EULAs (End User License Agreements) for software or software-based services that attempt to avoid responsibility for the software and services they sell
- Penalize companies with triple damages if they willfully disregard software best practices, lack adequate testing, ignore reported defects, or employ unqualified software developers
- Require certification and continuing education for software developers
- Empower regulatory panels to study software failures and make recommendations e.g., patterned after those that reviewed the Challenger or Titanic incidents
- Identify and adopt "best practices" widely, including for non-critical software systems like games

Other panelists pushed back on these suggestions.

Anita claimed that the software industry has improved steadily and it will continue to advance in many areas, such as better instrumentation of the development process. She went on, "I don't think tools are the only answer. But humans, tools, and AI, working together to address issues of scalability, composability, and complexity" will continue to make progress.

Priya was also more positive about the software industry than Landon. She recognized potential quality issues with the "democratization" of software engineering. While systems may be designed and built by experts, some applications may be developed by end users – so we must be aware of the relative quality differences. In the end, ongoing abstraction and the continuation of original agile principles will mean more people can solve problems using software principles appropriate to their own level of expertise.

Landon was critical of the "fail hard, fail fast" culture often found in the software industry, noting that this emphasis on failure has been an excuse for delivering poor quality software. It will be tragic if the "fail hard, fail fast and throw it out there and see what happens" culture is pervasive, and influences negatively (life threatening results) critical systems such as medical, avionics, and security systems.

Bertrand defended "fail fast" by explaining the agile thinking behind it. "When they say 'fail fast,' it's not that it is OK to fail. It's a different approach to program verification." He explained that agile development emphasizes frequent testing: "Their way of getting things correct is to test it all the time." Bertrand preferred other approaches like careful

up-front design and programming by contract, but he insisted that XP and other agile methods can build quality software with an iterative approach combined with a rigorous testing process: “I’m not ready to cast stones at the agile people... It’s not my approach, but it’s completely reasonable, and it is possible to combine it with other approaches.” Bertrand referred the audience to his recent book (*Agile: The Good, the Hype, and the Ugly*) for more details.

Kati explained that the software industry will need to address future issues that cannot be solved within software engineering alone. “We need to have a much better connection to psychology, sociology, neuroscience, environmental sciences, and so on.” In the past, when UX (user experience) was a new concept, “it brought a whole new perspective to many software developers.” Kati expressed that the broader view is critical: “to widen our horizons, think about the impact of software, and make conscious choices.” She explained, “I’m all for good engineering practices and teaching that as a science. But I don’t think it’s enough.”

## 5.2 The Future of Open Software

Bertrand questioned whether free software and open source software will continue as a major source of future innovation. He knew that criticism of the open source economic model is not a popular point of view. He warned that if you are too critical, “you appear to be a horrible representative of the powers of mercantile capitalism.” He admitted that the world has gained a lot from open source, but he pointed out that the experience of other engineering fields is unanimous: “there is no example of an engineering discipline which has gotten better in a context where there was no money to be made.” Bertrand was willing to consider open source to be “a different business model” that that is useful in some contexts, but the software engineering community needs to be realistic about its limitations.

Some economists view “free” software and services as a “barter” system – free products in exchange for user data. As a recent Harvard Business Review article [9] explained, “The business strategy of companies such as Facebook, Google, and numerous others is partly an exchange that does not entail money: Consumer data is being collected in exchange for the provision of internet services, just as berries might be swapped for meat.” The economics of open source may evolve along a similar path to the economics of social media websites and “free” online services such as Dropbox and Google Docs.

## 5.3 The Future of Software Engineering Technology and Education

Panelists suggested topics for inclusion in software engineering education and future software engineering technologies.

Landon advocated an approach to focus developers on building applications with an attention to correctness and reliability. “You have to start with the simplest cases [in education and industry]: trivial ‘Hello World’ programs, games, social media, and things that are not critical.” In computer science courses, it is better “to have assignments graded on whether they put in the appropriate amount of documentation and testing, not just on whether the program worked.” Landon advocated assessing assignments based on both development process and operational function.

Kati believed that ethical considerations are a fundamental consideration, particularly in AI applications and surveillance systems. She noted that society is exceptionally vulnerable since so much of our life is heavily reliant on software. “We need to start to think about the impact of what we do [with software] to others.”

Anita mentioned a study launched by SEI last year to anticipate the future of software engineering. That study, the National Agenda for Software Engineering Research & Development, proposed research focus areas and collected ideas on the importance of ethics in software. Other key observations included increases in automation, scalability, evolvability, and rapid deployment, as well as more applications of AI (such as AI-augmented software development).

Priya believed that the future of software engineering will need to combine technological advances with social components – technologies such as AI/ML and virtualization will assist in the democratization of development, build systems that incorporate empathy, and create software that embodies the characteristics of its users and makers. Priya saw this as a natural evolution of the path software engineering was already on.

Bertrand reflected on four influential technology areas that have been making their mark on software engineering over the last 30 years: object-oriented programming, open source, agile development, and cloud-related technologies (which includes microservices and DevOps). Bertrand saw the cloud-related technologies as positive so far (“it’s recent and the jury is still out”), but these techniques are changing how we do software engineering. The technologies cross some boundaries, because cloud, microservices, and DevOps are a blend of software engineering, networking, and systems administration, and they are likely to trigger big changes in application development. Bertrand’s view was that “traditional software engineering wisdom, principles, and modes of reasoning about the world do not completely transpose to that new world.”

## 6 Summary: Goals for the Future

The panelists shared their disagreements about the future of software engineering, but they agreed that technology, education, and markets will bring new challenges. The software community will find new ways to use AI, machine learning, remote working, cloud technology, natural language processing, and new software engineering tools to meet those new challenges.

Anita: The future will bring us smart automation, AI-inspired automation, evolving systems, composability and scalability of systems, and the architecture of new types of systems.

Priya: We will have a more inclusive software engineering ecosystem where everyone contributes. Building and integrating new software will be interdisciplinary, it will be ubiquitous, and it will be applied at different skill levels with varying ranges of expertise.

Landon: How the software industry integrates with society is key. We need continuing education – just because you are a software developer doesn’t mean you can stop learning. Practitioners will need to have certification and recertification – similar to other safety-critical medical and engineering fields.

Bertrand: Developers need make the right choices as they develop software technologies. Many choices of technology at all levels of the stack are influenced by extraneous

criteria: company policies, *a priori* tool selection, the color of the marketing brochure, or whatever.

Kati: Everybody needs to understand something about software development fundamentals, and about “good” software development. Fundamentals need to be part of every single curriculum at the university, especially for future leaders and managers.

Steve Fraser, the panel impresario, offered a pointer to a 2006 OOPSLA panel that had explored the “Future of Agile” [10]. Steve concluded the panel with the observation: “We need to remember the past in order to forge the future.”

## References

1. Ré, C.: Software 2.0 and Snorkel: beyond hand-labeled data. In: KDD 2018 ACM International Conference on Knowledge Discovery & Data Mining, p. 2876. <https://doi.org/10.1145/3219819.3219937> (2018)
2. Karpathy, A.: Software 2.0. <https://karpathy.medium.com/software-2-0-a64152b37c35>. Accessed 3 Jul. 2021 (2017)
3. France emergency service number disrupted after network outage, 3 Jun 2021. <https://www.bbc.com/news/world-europe-57341526>. Accessed 3 July 2021
4. Parnas, D.L.: Software engineering: a profession in waiting. *IEEE Comput.* **54**(5), 62–64 (2021). <https://doi.org/10.1109/MC.2021.3057685>
5. Shaw, M.: Research toward an engineering discipline of software. In: Proceedings of FoSER 2010: FSE/SDP Workshop on Future of Software Engineering Research, pp. 337–341. <https://doi.org/10.1145/1882362.1882431> (2010)
6. Shaw, M.: Progress toward an engineering discipline of software. Video of a talk for the SATURN 2015 conference, <https://www.youtube.com/watch?v=S03bsjs2YnQ>. Accessed 3 July 2021 (2015)
7. Cognizant: The future of work website. <https://www.cognizant.com/future-of-work>. Accessed 3 July 2021 (2021)
8. Carleton, A.D., Harper, E., Menzies, T., Xie, T., Eldh, S., Lyu, M.: The AI effect: working at the intersection of AI and SE. *IEEE Softw.* **37**(4), 26–35 (2020). <https://doi.org/10.1109/MS.2020.2987666>
9. Tett, G.: The data economy Is a Barter economy. *Harvard Business Review*, 6 Jul 2021 (2021)
10. Fraser, S., Rising, L., Ambler, S., Cockburn, A., Eckstein, J., Hussman, D., Miller, R., Striebeck, M., Thomas, D.: A fishbowl with piranhas: coalescence, convergence, or divergence? The future of Agile software development practices. In: Companion to OOPSLA ‘06, pp. 937–939. <https://doi.org/10.1145/1176617.1176750> (2006)



**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

