Editor: **Mik Kersten**
Tasktop
mik@tasktop.com

Five Predictions for the Coming Decades of Software

Mik Kersten

THIS DEPARTMENT HAS been exploring the state of the practice in DevOps, summarizing recent trends in scaling software delivery. For this issue of *IEEE Software* celebrating software engineering's 50th anniversary, I venture further ahead to consider how software engineering will evolve over the coming decades. My five predictions aren't intended to be precise; they aim to provide discussion topics for the shape of software engineering trends to come.

These predictions resulted from a celebration of the 50th anniversary of the University of British Columbia (UBC) Computer Science Department in May 2018. I participated in a panel of UBC alumni that discussed topics ranging from AI's future impact to where computer science students should focus today's studies to have the best job prospects.

During that panel, I made a case for using an economic model of technological cycles when considering the decades ahead. In *Technological Revolutions and Financial Capital*, Carlota Perez proposes that in the last three centuries we've experienced five "great surges of development."[1] Other Kontratiev-wave or "long wave" theories continue to be debated for modeling the diffusion of new technological waves. Perez's model is unique in specifying that each wave consists of an *Installation Period* and a *Deployment Period*. During the Installation Period, a critical factor in production becomes cheap, and infrastructure is built to exploit it. During this period, a constellation of innovation, entrepreneurship, and financial capital drives a surge. Next comes a disruptive Turning Point, followed by a Deployment Period of stabilization that results in widespread productivity gains for the new technology.

During a conversation I had with Perez in June, she stated her belief that we're still at the Turning Point. Although it's impossible to know today whether the pattern Perez describes will repeat itself, I'll use the model to make some predictions here. In particular, the Turning Point theory provides historical examples of the kinds of disruption that DevOps and agile are trying to address. This is the case I make in my upcoming book *Project to Product*, which discusses the Turning Point's effects on technology organizations that need to scale their software delivery practices to survive the next decade.[2] Consequently, my predictions are based on how software engineering will need to change as we shift from this Turning Point to the Deployment Period.

An Age of Invention Will Pivot to an Age of Adoption

During each of these surges, the bulk of the groundbreaking invention occurs during the Installation Period. This is when the combination of a new technological system and entrepreneurship fuels invention and is exploited by new companies that become the technology giants of their age. In the last age, examples of those were the car companies that mastered mass production. In the current age, the means of mass production is software, and the FAANG (Facebook, Apple, Amazon, Netflix, and Google) companies have risen as they've mastered software delivery at scale. During the Deployment Period, if what we witness is similar to the past four ages, over the coming two decades, the rest of the economy will slowly start to see similar productivity gains as software practices are adopted and mastered across industries and organizations.

The question then becomes, what's different about software engineering in the Deployment Period compared to the Installation Period?

0740-7459/18/$33.00 © 2018 IEEE

We can expect to shift to more incremental innovation and a slowing pace of new breakthroughs in our field. For example, it's unlikely that we'll create new programming-language paradigms that dramatically increase developers' productivity, as we saw with the introduction of object-oriented programming. Instead, we can expect more refinement and cross-pollination of existing ideas—for example, as we've seen with the marriage of object-oriented runtimes and SDKs (software developer's kits) with dynamic programming and Lisp via Clojure. The Perez model also predicts that candidates for the next technological wave start arising during the Deployment Period. If that holds true, today's research will be creating the next wave's technological foundations. But notably, the new wave will grow largely independently of the ongoing diffusion of this one.

Given that software engineering is headed into a diffusion phase, opportunities will grow for deploying the ideas and innovations of the Installation Period into existing and new business models at a much greater scale than in the past 50 years. For example, Mark Weiser, Xerox PARC's Chief Scientist at the time, coined the idea of ubiquitous computing in 1988. Although the conceptual foundation for the Internet of Things (IoT) was laid down at that point, only now do we have the computational resources and business ecosystem for the IoT to become a key part of the world's digital infrastructure.

Software Complexity Will Drive Specialization

The computational ideas formed in the past 50 years will become much more broadly adopted over the next 50 years, and in the process will be tailored to each segment of the economy. To support this, the principles of software engineering will need to spread much more broadly and deeply within businesses. Software is already becoming a key competitive differentiator for business offerings ranging from financial services to natural resources. It is already the norm for large banks to have tens of thousands of IT staff and for oil companies to manage portfolios of hundreds of internal software applications. With software becoming more ubiquitous in relation to the means of production and competitive differentiation, computational thinking will become key to many professional and business functions.

As this happens, the nature of programming will evolve. We're already seeing increasing specialization of programming expertise up and down the software stack. In the past, programmers tended to be trained as generalists. Now, the sophistication of today's software and toolkits has driven a growing specialization of programmers, ranging from user interface developers who have never pondered the code for a TCP/IP stack, to machine-learning specialists who cringe at the idea of following the fashion-like trends such as JavaScript frameworks.

The specialization goes beyond the code. Some Internet-scale software vendors split their software delivery staff evenly between development and operations, foreshadowing the increasing importance of infrastructure engineering. As infrastructure continues to turn into code, specialized operations roles are emerging. Add to that the specializations in design, business analysis, and security, and a growing number of subspecializations specific to industry domains such as automotive or healthcare software. Whereas the past decades of software engineering created innovators and generalists, the coming ones will be all about specialists with deep domain and functional expertise.

As Automation Grows, so Will the Demand for IT Professionals

In medicine's early days, treatment was likely to be administered by a single general practitioner. Today, the Association of American Medical Colleges lists more than 120 specializations and subspecialties.[3] The growing specialization in how software is built will require increasingly specialized education programs and training, as well as increasing breadth of practices and tools. Just as we're seeing programming languages, frameworks, and tools specialize for specific functions, such as machine learning and web development, we can expect further specialization of programming paradigms.

Some domains—for example, those with well-defined data models—will be outsourced to AI sooner than others. We're already seeing this happen in image recognition and natural-language processing. In a growing number of data-driven domains, AI will become better at some aspects of software engineering than human programmers. However, even as AI and automation increase dramatically, the number of humans involved in software delivery will grow. This will be due to the spread of software into every industry, the growing specialization of roles, and the shift of business processes to software. Whether the software is being created by programmers or AI, the number of supporting specializations and the demand for those specialists will continue to grow throughout the Deployment Period.

Coding Will Evolve into Domain Expertise and Data Modeling

As the adoption of AI grows, the way that some software is built will change. The human-communication-intensive aspects, such as designing and defining requirements or user experiences specific to business processes, are likely to still require growing numbers of people. However, for domains suited to ongoing AI developments, some traditional programming work will shift to data modeling. Similarly, data science roles will continue to grow in data-intensive domains.

Computation, processing, and storage prices will continue to drop; capabilities will continue to increase. Dramatic advances, such as quantum computation, could take hold. As that happens, new jobs and specializations will emerge around producing and analyzing the results of increasingly massive computational and data capabilities. This will then need to be fed back into the creative process of defining business strategies and producing new customer offerings.

AI Will Become Its Own Field of Practice

If the Perez model of technological revolutions holds true, once we're through the Turning Point, we should see candidates for the next wave arise and then mature over the coming decades to create that wave of innovation and disruption. If such a sixth wave is currently forming, we can imagine it to be as profound as this one.

Computer science and software engineering were forming in universities' mathematics and electrical-engineering departments 50 years ago. If the pattern is to repeat, offshoots of today's software engineering work will define both a new university department and the disruptive forces for the next 50 years. Will that new department be built around general or strong AI (that is, artificial general intelligence)? Signs of that are appearing, for example, with Carnegie Mellon University's Machine Learning Department (see https://www.ml.cmu.edu). Or, will computation and AI become so pervasive as to become a core part of every department and business practice? Time will tell.

For now, as the practices we've honed and practiced over the past 50 years become embedded across the economy, our role is to support those growing specializations. And, to honor the ethical aspects of engineering disciplines, it's our responsibility to help provide the ethical guidelines to our colleagues embarking on the journey of creating the next wave.

References

1. C. Perez, *Technological Revolutions and Financial Capital: The Dynamics of Bubbles and Golden Ages*, Edward Elgar, 2003.
2. M. Kersten, *Project to Product: How to Survive and Thrive in the Age of Digital Disruption with the Flow Framework*, IT Revolution Press, 2018.
3. "Careers in Medicine," Assoc. Am. Medical Colleges, 2018; https://www.aamc.org/cim/specialty/exploreoptions/list.

ABOUT THE AUTHOR

MIK KERSTEN is the Founder and CEO of Tasktop. Contact him at mik@tasktop.com or follow @mik_kersten.