DOI:10.1145/3640316

George V. Neville-Neil

Article development led by acmqueue
queue.acm.org

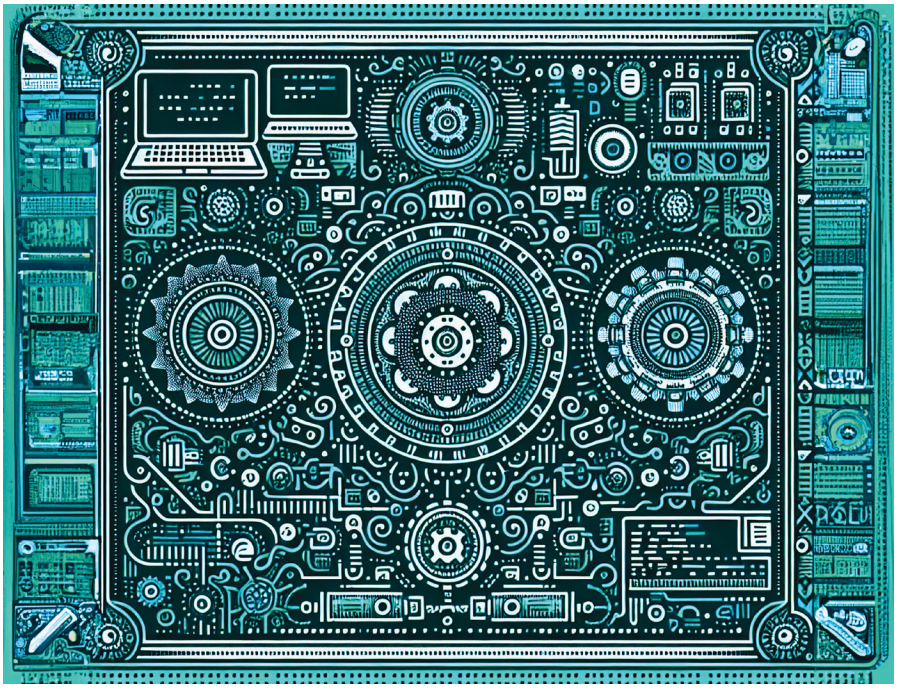
Kode Vicious
Is There Another System?

Computer science is the study of what can be automated.

Dear KV,

There has been considerable discussion about the use of AI for automating a lot of work—including software development. I have just finished a Ph.D., I am out of school, and I have been working for a few years. But now I wonder if software research and development will have a future that includes people like me, or if I will simply be automated out of a career.

Colossally Concerned



Dear CC,

It might be odd to think that someone with an advanced degree could be automated out of a job, but I live in New York City, where there are plenty of minimum-wage workers with advanced degrees. I am afraid we are yet again caught in another tech hype cycle around an advance in software, and that is causing a lot of confusion both inside and outside our industry.

Do large language models pose a threat to software developers? That depends on what kind of software they develop and how easy it might be to automate their work, but this has always been true. Computer science is the study of what can be automated, and your corporate masters see your salary as impinging on their bonuses, so they are always happy to reduce the number of human resources.

Computer science and software development change more quickly than many other fields because what we do does not require much in the physical realm and because, for the moment, the fallout from our mistakes goes mostly unregulated and unpunished. Consider what it takes to innovate the construction of buildings or other physical infrastructures and you'll get what I mean. New bridges are built with newer materials, but such changes take years or even decades, while a new fad in computing can sweep at least part of the field in a few months. Like cryptocurrency and the Internet bubble before it, the "AI" bubble, and I put "AI" in quotes because—as a good friend said recently—"AI is what people say when a computer does something they thought only a human could do."

Can large language models replace some parts of software development? Perhaps. I have seen evidence both for and against, and the Internet is littered with arguments on both sides.

It occurs to me that KV answered this question in another form many years ago[a] when I discussed how to stay up to date and fresh on the latest changes in computing. The key to a long career—as is obvious to those who have watched KV babble on for many years—is to continue to survey the field, see what is new, try new

a See https://bit.ly/3OXkc59 and the second letter (Bummed) at https://bit.ly/3SQ1EWM

IMAGE CREATED BY ANDRIJ BORYS ASSOCIATES WITH THE ASSISTANCE OF SHUTTERSTOCK.AI

things, and see what works well for you.

KV has yet to see evidence of general AI appearing and replacing people, although the hype machine keeps telling us it is just around the corner, so you need to think of these new systems as aids to the programmer, much as early compilers were in the 1960s and 1970s.

Back when the first compilers appeared, they produced machine language that was not nearly as efficient as what was created by working programmers. Today, there are only a few of us who understand or work in assembly or machine code, and this is both good and bad. It is good because it means that most programmers can express concepts in code that would have been tortuous to produce on earlier systems. It is bad because machines still run machine code, and if you cannot debug it, often you cannot find the true source of a performance or other issue. Compilers are tools, debuggers are tools, large language models are tools, and humans are—for the most part—toolmakers and users.

Can large language models replace some parts of software development?

One of the easiest tests to determine if you are at risk is to look hard at what you do every day and see if you, yourself, could code yourself out of a job. Programming involves a lot of rote work—templating, boilerplate, and the like. If you can see a way to write a system to replace yourself, either do it, do not tell your bosses, and collect your salary while reading novels in your cubicle, or look for something more challenging to work on.

There are days—I should say mostly late, late nights—when KV wishes the machines would take over. I would gladly be a battery if I could just have some peace and quiet to think about the higher-order things in computer science, algorithms, operating systems, and efficiency. These creative endeavors are still beyond the reach of whatever it is we call "AI," and KV is willing to wager they will remain so for the duration of his—and your—career.

KV

Related articles on queue.acm.org

AI Gets a Brain
Jeff Barr and Luis Felipe Cabrera
https://queue.acm.org/detail.cfm?id=1142067

Improving Testing of Deep-learning Systems
Harsh Deokuliar, Raghvinder S. Sangwan, Yoaukim Badr, Satish M. Srinivasan
https://queue.acm.org/detail.cfm?id=3631340

Testable System Administration
Mark Burgess
https://queue.acm.org/detail.cfm?id=1937179

George V. Neville-Neil (kv@acm.org) is the proprietor of Neville-Neil Consulting, Brooklyn, NY, USA, and co-chair of the *ACM Queue* editorial board. He works on networking and operating systems code for fun and profit, teaches courses on various programming-related subjects, and encourages your comments, quips, and code snips pertaining to his *Communications* column.