BLOG@CACM

The *Communications* Web site, http://cacm.acm.org, features more than a dozen bloggers in the BLOG@CACM community. In each issue of *Communications*, we'll publish selected posts or excerpts.

twitter
Follow us on Twitter at http://twitter.com/blogCACM

DOI:10.1145/3213760 http://cacm.acm.org/blogs/blog-cacm

We Are Done with 'Hacking'

Today's programmers offer more valuable skills than simply being able to hack algorithms and make data structures, says Yegor Bugayenko.

Yegor Bugayenko
The Era of Hackers Is Over
http://bit.ly/2JH8qrg
April 23, 2018

In the 1970s, when Microsoft and Apple were founded, programming was an art only a limited group of dedicated enthusiasts actually knew how to perform properly. CPUs were rather slow, personal computers had a very limited amount of memory, and monitors were lo-res. To create something decent, a programmer had to fight against actual hardware limitations.

In order to win in this war, programmers had to be both trained and talented in computer science, a science that was at that time mostly about algorithms and data structures. The first three volumes of the famous book *The Art of Computer Programming* by Donald Knuth, a Stanford University professor and a Turing Award recipient, were published in 1968–1973. This programming bible earned a famous comment from Bill Gates: "It took incredible discipline, and several months, for me to read it."

This comment demonstrates the reality that creating a simple piece of software was a complex engineering task, even though software engineering only "emerged as a discipline in its own right" later on in the 1980s, as Ian Sommerville said in his 1982 book *Software Engineering* (https://dl.acm.org/citation.cfm?id=1841764).

Most programmers were calling themselves "hackers," even though in the early 1980s this word, according to Steven Levy's book *Hackers: Heroes of the Computer Revolution*, "had acquired a specific and negative connotation." Since the 1990s, this label has become "a shibboleth that identifies one as a member of the tribe," as linguist Geoff Nunberg pointed out (https://n.pr/2rg8g2v).

Being able to hack both the software and the hardware was a virtue for a long time. However, the world of computer programming has changed dramatically in the last decade.

First of all, the cost of computing power gets cheaper every year. For example, one gigabyte of computer memory cost about $1,000 in 2000; in 2018, it costs less than $5. That is 200 times cheaper in the span of only 18 years. The same is true for hard drives, monitors, CPUs, and all other hardware resources. As James Somers noticed in his analysis of industry problems in *The Coming Software Apocalypse* (https://theatln.tc/2HFazak): "Computers [have] doubled in power every 18 months for the last 40 years."

Second, the growth of open source is massive. The majority of software is available for free now along with its source code, including operating systems, graphics processors, compilers, editors, frameworks, cryptography tools, and whatever else we can imagine. Programmers do not need to write much code anymore; all they need to do in most cases is wire together already available components.

Third, despite an increasing population of programmers in the world, the field is still in a deficit. In some European countries, the demand for highly skilled IT personnel is twice as high as their market supply of talent. According to Iamexpat.nl, in the Netherlands "a whopping 76% of HR employees reported hav-

ing difficulty finding enough candidates with this qualification."

Fourth, programmers now work remotely instead of in offices or cubicles. Thanks to the growth of high-speed Internet, conferencing software like Zoom and Skype, messaging tools like Slack and Telegram, and distributed repository managers like GitHub and Bitbucket, along with many other innovations, remote work has become more comfortable than the alternative of working in the traditional office setting.

Finally, programmer salaries have skyrocketed in the last few decades. In 2000, when one gigabyte of memory still cost $1,000, the average senior programmer earned around $80,000 in Silicon Valley. In 2018, they are currently making three times more, while RAM is 200 times cheaper.

Taking these five variables into account, it would appear that the skills required of professional and successful programmers are drastically different from the ones needed back in the 1990s. The profession now requires less mathematics and algorithms and instead emphasizes more skills under the umbrella term "sociotech." Susan Long illustrates in her book *Socioanalytic Methods: Discovering the Hidden in Organizations and Social Systems* (http://bit.ly/2w0sFhS) that the term "sociotechnical systems" was coined by Eric Trist et al. in the World War II era based on their work with English coal miners at the Tavistock Institute in London. The term now seems more suitable to the new skills and techniques modern programmers need.

They need to know how to communicate with the open source community to find the needed components, to request features, and to learn bug fixes from their developers. Moreover, they have to be ready to contribute to open source software by submitting pull requests or even creating their own programs. Those who used to work only with commercial and private software will soon be far behind other programmers.

They have to know how to get help outside of an office or even a project team when working remotely and alone. Aside from Stack Overflow, which dominates the Q&A platform for programming market, there are documentation and code repositories that a professional programmer must know how to navigate. Those who previously only relied on colleagues and friends will now lose to those who know how to learn from the entire Internet.

Programmers have to know how to write maintainable code that other programmers will be able to easily understand. Since hiring personnel grows more expensive every year, businesses emphasize the maintainability of their code bases over developing exceptionally complex code. It is easier for them to buy a larger server if the algorithm is not fast enough, rather than lose what previous programmers created when a new team or a replacement shows up and fails to understand how to modify the project. Because the cost of computers continues to grow cheaper and the cost for employing programmers continues to increase, maintainability continues to dominate the programming landscape as the primary virtue of almost any software. The end result is that these "Hackers" who spend their days writing complex, cryptic code will soon find themselves out of the market.

Edsger W. Dijkstra's words—"Simplicity is a great virtue"—which he uttered in 1984, grow increasingly more valuable every year.

It seems that the future of programming rests less in math and more in sociotech relationships between people.

Comments

Not sure I agree with the viewpoints in this blog. "Hacker" is a casually understood term, it could be simply understood as someone who is extremely sophisticated in writing code, but not necessarily isolated from engaging in sociotech relationships. I am sure that "hackers" do run into mental blocks, and they, too, look for answers at "Stack Overflow" or happily provide some to "show off" their "hacking" skills. "Hacking" doesn't necessarily make code more complex than necessary, it may do just the opposite — making code simpler than necessary that affects readability. "Hacking" might also mean the "code-and-fix" practice, which the article is right about. But the verdict is just as old as the term "software engineering"—we need software craftsmanship, if not an engineering process, to build software. "Hackers" are always there regardless of how we define them, and I am sure that their productivity can be turned into good use.

Does the future of programming rest less in math? Coding more efficient algorithms (regardless of RAM price dropping) or writing more efficient code requires mathematical thinking. Had our developers known Z-Specification (a formal specification method based on Set Theory), they would have written much more reliable software than we have experienced. The future of programming may be dominated by data-analytics, machine-learning and deep-learning applications. Developers are not just to collect data and hit a button to invoke "shop-provided" models; rather, they are to tweak the models and, by all means, develop new models to be trained by datasets unique to the local business environment. To the opposite, the future of programming rests more in mathematics. After all, computing, by its very nature, is a mathematical discipline.

—Chenglie Hu

I, too, would have to disagree. The hardware of computers today has changed so drastically from those early days that to really write proficient/optimized code, a software engineer has to really understand the hardware and what it is doing … so "Hackers" live, especially in the fields where speed and efficiency are very important. One has to understand memory, caches, cores and hyper-threading, throughput and latency, vector registers, and how to write code to properly take full advantage.

In order to get that mathematics you write of to run efficiently, one has to know how the computer handles the calculations, fetches, executes, and stores the results the best and fastest way.

Today more than ever. software engineers need to be "Hackers" in the manner that Steven Levy referred to them.

—Rod Haxton

Yegor Bugayenko is founder and CEO of software engineering and management platform Zerocracy.