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.

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

DOI:10.1145/3204443

<http://cacm.acm.org/blogs/blog-cacm>

Programming Programming Languages, and Analyzing Facebook's Failure

Mark Guzdial considers an idea with significant educational implications, while Susan Landau looks into the Cambridge Analytica/Facebook scandal.

Mark Guzdial
A Computing Education Research Perspective on Programmable Programming Languages

<http://bit.ly/2JcSr4T>

March 8, 2018

When the March *Communications* appeared in my mailbox, my monthly Blog@CACM post wrote itself. The cover story, an important idea with significant education implications, was *A Programmable Programming Language* by Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, Shriram Krishnamurthi, Eli Barzilay, Jay McCarthy, and Sam Tobin-Hochstadt. The authors describe their work with Racket, where software developers redesign the programming language to match a problem being solved.

In the ideal world, software developers would analyze each problem in the language of its domain, then articulate solutions in matching terms. They could

communicate with domain experts and separate problem-specific ideas from the details of general-purpose languages and specific program design decisions.

By “language,” we mean a new syntax, a static semantics, and a dynamic semantics that usually maps the new syntax to elements of the host language and possibly external languages via a foreign-function interface (FFI).

The article does a good job of playing out the issues and possibilities. I’m particularly taken with the explanation of how types play an important role in specifying the new languages. The authors have thought a lot about how a language can usefully constrain and facilitate programmers’ work to improve problem-solving and productivity.

To fit into *CACM*, the paper is necessarily short, so not all the issues are laid out. I know from discussions with Krishnamurthi these ideas *started* from their work in education with DrRacket. The team wanted to define subsets of the

language that were easier to teach, and generalizing that idea led to this work.

Programmable programming languages have been created in the past. Smalltalk-72 objects could re-parse their input; the language included an “eyeball” character in methods that could look ahead in the token stream to parse the rest of the input line. Dan Ingalls wrote in the “Smalltalk-80: Bits of History, Words of Advice” that this made Smalltalk-72 difficult to grow. The programmer was challenged to design a new language that could be understandable by others. How do programmers afterwards know the context and thinking of the programmer who implemented this language? How do they learn “Smalltalk-72” when it’s really an umbrella for a bunch of different languages? This modern work is richer for thinking about developing new programming languages for particular problems, but the educational issues are still there.

Programming is not just about problem-solving.

► *Programming is a job skill.* It is hard for students to learn programming. Students often learn a particular programming language to get themselves jobs. Are problem-specific programming languages easier to learn? Will some become so popular and useful that it’s worthwhile (from a job perspective) for a student to learn a problem-specific programming language?

► *Programming creates infrastructure.* Programming is the infrastructure for our world. There are large systems still in use today written in Cobol and PL/1, and we have to maintain that informa-

tion infrastructure. Does creating more languages in problem-specific forms make it harder to find enough programmers who know infrastructure built in these problem-specific forms? Is it harder or easier to train new programmers to maintain that infrastructure?

► *Intellectual property is embodied in programming.* Programs are intellectual property. When intellectual property is defined in terms of thousands of lines of code, IP can't be easily carried away in someone's head. But when the IP is in the *language*, which can be learned and carried in a single person's head, the definition of what's protected and what was just *learned* seems complicated. If a programmer moves from Company A to Company B, and that programmer has learned the problem-specific language in Company A, and then re-implements it in Company B, was intellectual property stolen?

I love that this article makes all of software soft again; everything is malleable, down to the programming language itself. As an education researcher, I know learning programming is a struggle. New problem-specific languages may increase the challenges for computing education research. Because of the value of these kinds of languages, the new research questions raised are worth investigating.

Susan Landau
What Went Wrong?
Facebook and
'Sharing' Data with
Cambridge Analytica
<http://bit.ly/2uFPjv3>

March 28, 2018

The road to the Cambridge Analytica/Facebook scandal is strewn with failures. There's the failure to protect users' privacy, the failure to protect voters, and the failure to uncover the actions and violations of laws that may well have affected the Brexit referendum and the U.S. presidential election. The latter two threaten the heart of democracy.

I want to focus on the failure to protect users' privacy, which has the virtue of being easy to unpack, even if its resolution is far from simple. This failure to protect privacy has multiple parts.

First, there's Facebook's failure to protect user privacy. Indeed, the company's aim was quite the opposite. Facebook believed "every app could be social." That meant giving broad access

not only to a user's data, but also to that of his "friends." In 2013, Cambridge University researcher Aleksandr Kogan paid 270,000 Facebook users to take a personality quiz (the money came from Cambridge Analytica, but that's another part of the story). Doing so gave Kogan's app the ability to "scrape" information from their profiles. In those days, Facebook's platform permitted the app not only to access the quiz takers' profiles and "scrape" information from them; the social network also allowed apps to do the same to the profiles of the quiz takers' "friends"—50 million of them.

Data from the friends would be collected unless the friends explicitly prohibited such collection. The ability to do so was not easy; users were not told their data would be shared if a Facebook friend engaged an app that did such scraping. To prevent collection, users had to first find out that their friends' apps would do this, then configure their profiles to prevent such data sharing.

Then there's Facebook's failure to take legal action after the company became aware the data of those 50 million Facebook users had been provided to Cambridge Analytica. This data transference violated Kogan's agreement with Facebook in acquiring the data. When Facebook found out, it requested Cambridge Analytica certify they had destroyed the user files; the Silicon Valley company did not ensure Cambridge Analytica had done so. As we know, Cambridge Analytica did not comply. Facebook's failure to ensure the files were destroyed was failure number 2.

Finally, there's Facebook's failure to inform the 50 million users whose data was taken. There was a breach of contract between Kogan and Facebook, but there was also a privacy breach: the profiles of 50 million Facebook users were being used by Cambridge Analytica, a British firm specializing in using personal data for highly targeted, highly personalized political ads. Facebook failed to inform those 50 million users of the breach. That was failure number 3.

Facebook is on the way to repairing some of these failures. In 2014, Facebook limited the information apps could collect on Facebook friends—though not fully. Mark Zuckerberg said Facebook will, belatedly, inform the 50 million Facebook users of the privacy breach that happened in 2014.

But there are other failures as well.

The fourth is society's, which hasn't been taking privacy particularly seriously. This isn't universally true. In the U.S., for example, the Federal Trade Commission (FTC) and the states' Attorneys General have taken companies to court when the firms fail to protect users' privacy or fail to follow their own privacy policies. But their set of tools for doing so is quite limited. There are a handful of laws that protect privacy in particular sectors. There are fines if companies fail to live up to stated privacy policies. There are data breach laws. And there's the ability to fine if actual harm has been caused.

The Facebook/Cambridge Analytica case is garnering significant attention by both the FTC and the states' Attorneys General. It helps that in 2011, the FTC acquired a powerful tool when Facebook signed a consent decree as a result of an earlier privacy violation, which required the company to make clear "the extent to which [Facebook] makes or has made covered information accessible to third parties." Did the fact that those 50 million "friends" had difficulty preventing collection of their data constitute a violation of the consent decree? We will find out. At a potential \$40,000 per violation, the consequence could be quite expensive for Facebook.

There's a fifth failure that may well be most important of all: our willingness to trade data about ourselves for seemingly free services. That's nonsense; services that manipulate how you spend your time, how you feel, and whom you vote for are anything but free. Maybe it's time to cut that connection? Some things will be harder to lose—seeing that photo of your nephews clowning at breakfast, or getting updates from the folks at your previous job—but you may find an extra hour in your day. That's an hour to call a friend, take a walk, or read a book. It sounds like a good trade to me. I wouldn't actually know; I value my privacy too much, and never joined the network in the first place.

Mark Guzdial is a professor in the Computer Science & Engineering Division, and jointly in the Engineering Education Research program, of the University of Michigan. Guest blogger **Susan Landau** is a computer scientist and cybersecurity policy expert at the Fletcher School of Law & Diplomacy and the School of Engineering, Department of Computer Science, Tufts University.

© 2018 ACM 0001-0782/18/6 \$15.00