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:10.1145/2534706.2534710 http://cacm.acm.org/blogs/blog-cacm

The Lure of Live Coding; The Attraction of Small Data

Mark Guzdial ponders a new set of research questions, while Valerie Barr considers the utility of one person's data.

Mark Guzdial
"Trip Report on Dagstuhl Seminar on Live Coding"
http://cacm.acm.org/blogs/blog-cacm/168153-trip-report-on-dagstuhl-seminar-on-live-coding/fulltext
September 26, 2013

I recently spent time at the remarkable Schloss-Dagstuhl in a seminar on "Collaboration and Learning through Live Coding," organized by Alan Blackwell, Julian Rohrhuber, and Alex McLean (and James Noble, who was not able to join us). It was a terrific event that has me thinking about a whole new set of research questions.

Live coding is a performance where programmers improvise music (and sometimes visuals) in front of an audience. It is real-time coding to create real-time music. Often, this occurs in pairs, trios, or even larger groups (see a quartet at http://www.youtube.com/watch?v=jlx6KNo5Eok). One programmer starts his music going, then the other one writes some code to respond and interact, and they both go off. It is a jam session on laptops.

The main live coding website is TOPLAP.org. There is another website devoted to AlgoRave—dance parties, driven by live coders generating the music. I still find that a little strange to think about: people dancing to some hacker's algorithms. For an introduction to live coding, I recommend this video from Andrew Brown (at http://vimeo.com/74081305), who wrote the JMusic library for music composition in Java that we use in our Media Computation curriculum, and also this video from Sam Aaron (at http://vimeo.com/22798433) on live coding in Overtone. Sam Aaron also created Sonic Pi, which is a synthesizer programming environment for the Raspberry Pi.

Most of the attendees were live coders, but there were a number of us others who helped explore the boundary disciplines for live coding. The seminar explored the ramifications and research potential of this activity.

Robert Biddle was there to lead discussions about the software engineering implications of live coding. On the one hand, live coding feels like the antithesis of software engineering, or as one attendee put it, "an ironic response to software engineering." There is no test-driven development, no commenting, no development of abstractions (at least, not while live coding), no exploration of user needs. On the other hand, live coding (not necessarily with music) can be an important part of exploring a space. One could imagine using live coding practices as part of a conversation with a user about needs and how the programmer understands those needs.

Geoff Cox led a conversation about the humanities research directions in live coding. Geoff has a particular interest in labor, and he pointed out how live coding surfaces hidden aspects of the labor in modern society. While computing technology has become ubiquitous in the developed world, few people in our society have ever seen source code or a programmer. What does it mean for an audience to see an

embodiment of a programmer, to see the labor of generating code? What is more, the audience is seeing code doing something that is not normally associated with people's notions of what code is for—making music. How does this change the audience's relation to the computing technology? The notion of programming-as-performance is an interesting and novel way of thinking about computing practice, and in sharp contrast to stereotypical perspectives of programming.

Thomas Green, Alan Blackwell, and others from the PPIG (Psychology of Programming Interest Group, http://www.ppig.org) community pointed to the notations that the live coders used. I have drawn on Thomas' work on the cognitive dimensions of programming and the implications of our programming representations since I was a graduate student. The language constraints for live coding are amazing. Live coders tend not to use traditional development languages like C++ or Java, but instead work in Scheme, Haskell, and a variety of domain-specific languages (like SuperCollider)—often building their own implementations. Live coders need languages that are expressive, provide for the ability to use techniques like temporal recursion, are concise, and (as one live coder put it) "let me live code at 2 A.M. at a dance club after a couple of beers."

I was there to connect live coding to computing education. I learned the connections from the seminar—I hadn't really seen them before I got there. I am fascinated by the humanities questions about introducing source code and programmers to a technologically sophisticated audience that probably never saw the development of code. I am also interested in the value of rapid feedback (through music) in the performance. Does that help students understand the relationship between the code and the execution? How does the collaboration in live coding (for example, writing music based on other live coders' music) change the perception of the asocial nature of programming?

Live coding is rich with interesting research issues because it is exploring such a different space than much of what we do in computer science. It is about expression, not engineering. It is about liveness, not planfulness. It is about immediate creation of an experience, not later production of an artifact. That makes it worth exploring.

Valerie Barr "The Frontier of Small Data"

http://cacm.acm.org/blogs/blog-cacm/168268-the-frontier-of-small-data/fulltext

September 29, 2013

I had the opportunity recently to attend a talk by Deborah Estrin, a professor of computer science at Cornell Tech in New York City, entitled "small, n = me, data." Her title is a play on our usual way of referring to problem size; in this case, making the "n" of the problem size just a single person. Certainly Estrin is not turning her back on big data and all that can be learned from it, but she is very interested in how a single person's data can be used to understand their situtation. For example, in the medical realm, the Open mHealth effort is developing an architecture that can integrate data from an individual's use of specific apps in order to help a health care provider make recommendations.

Particularly interesting is the small data effort that Estrin has undertaken at Cornell NYC Tech. Her definition of small data is "your row of their data." What observations would surface if you could analyze in some combined way your mobile usage, cable usage, utility usage, e-commerce activities, search activities, social media and email usage, automobile usage gathered from smart car data, use of games, music, and video? What changes in the health and well-being of an aging parent might surface through analysis of their aggregated data? Would one be better able to compare the efficacy of different courses of medical treatment by looking at aggregated data? For example, data traces showing how far you walked and how early in the morning you left the house could indicate relative effectiveness of an arthritis medication.

Estrin headed off possible issues with the comment "where there is a privacy concern, there is an opportunity." Her goal is to develop an "ecosystem of applications" that an individual can run over her or his own set of data streams, a collection she referred to as our "personal data vault." Her hope is that eventually we will be able to subscribe to our own individual data traces. Personal data APIs will allow for development of real-time personal data apps.

Estrin listed several key challenges:

1. getting the data
2. processing and making sense of noisy, diverse data
3. secure models for the personal data vaults
4. a testbed for app prototypes

You can read Estrin's 2013 TEDMED talk online (at http://smalldata.tech.cornell.edu/narrative.php) to get more information. She closed the talk I attended with a reminder that Cornell NYC Tech is actively recruiting graduate students, so pass along the link to their Admissions page (http://tech.cornell.edu/admissions/). I know I definitely have students who will be very interested in the possibility of working on this small data project.

Mark Guzdial is a professor at the Georgia Institute of Technology. **Valerie Barr** is a professor at Union College.