The Ongoing Computer Revolution

Butler Lampson

Talk at the Draper Award presentation, Washington, DC, February 2004
Published in *The Bridge*, **34**. 2 (Summer 2004)

I remember reading about the Draper prize a few years ago and thinking "Wow, they give that for inventing the jet engine or communications satellites or the integrated circuit." It's a great honor to join that august company tonight.

I studied physics in college, and I went to graduate school at Berkeley to study more physics. But I started programming in high school on an IBM 650, and it was always a big distraction from physics. At Berkeley I was lucky enough to stumble across an Arpa project, hidden behind an unmarked door, that was building one of the first time sharing systems; that system later became a commercial product, the SDS 940. I was immediately seduced, abandoned physics, and never looked back (which was fortunate considering what the job market was like for new physics Ph.D.s five years later).

Since then, after brief stints on the Berkeley faculty and at a failed startup, I've worked for three major computing research laboratories: Xerox PARC's Computer Science Laboratory, Digital's Systems Research Center, and Microsoft Research.

I worked for Bob Taylor, directly or indirectly, for 30 years, from my time at Berkeley until I joined Microsoft. I have unbounded admiration for his two outstanding qualities:

- His ability to attract outstanding people and coax them into working as a whole that is greater than the sum of its parts.
- His judgment about what research problems are both truly important and ripe for solution. Roger Needham said that the secret of success in computer systems research is to attack problems that you know are easy, but that everyone else thinks are too hard, and that's always what we did in Bob's research programs.

At PARC we stood on the shoulders of giants, the people who invented time-sharing, the Arpa and Aloha networks, programming languages, operating systems, Doug Engelbart's on-line system and his mouse, Sketchpad. Nearly all of us came from the community of researchers fostered by Arpa.

Xerox asked us to invent the electronic office, even though no one knew what that meant. We did, and everyone's using it today. That makes it hard to remember what the world was like in 1972. Most people thought it was crazy to devote a whole computer to the needs of one person—after all, machines are fast and people are slow. But that's true only if the person has to play on the machine's terms. If the machine has to make things

comfortable for the person, it's the other way around. No machine, even today, can yet keep up with a person's speech and vision.

The most important property of what we built is that it's universal (within its domain, of course). Everyone knew that a computer can compute anything, but in the Alto system the screen can display any image (well, on the Alto it has to be black and white), the printer can print any image, the network can communicate anything, and the software lets you construct anything. Perhaps not quite anything, but you can do typeset text, drawings, pictures, music, animation, voice. Of the things you can do today with computers, we missed numbers, because we didn't have much personal use for spreadsheets, and databases and mathematics were too hard.

The system is enabled by the hardware conceived, designed, and built by Chuck Thacker. In spite of that, the software is everything, even though it's nothing tangible. With software, you can make the system do anything. This is even more true today.

> There's a story about some people who were writing the software for an early avionics computer. One day they get a visit from the weight control officer, who is responsible for the total weight of the plane.
>
> "You're building software?"
>
> "Yes."
>
> "How much does it weigh?"
>
> "It doesn't weigh anything."
>
> "Come on, you can't fool me. They all say that."
>
> "No, it *really* doesn't weigh anything."
>
> After half an hour of back and forth he gives it up. But two days later he comes back and says, "I've got you guys pinned to the wall. I came in last night, and the janitor showed me where you keep your software." He opens a closet door, and there are boxes and boxes of punch cards. "You can't tell me those don't weigh anything!"
>
> After a short pause, they explain to him, very gently, that the software is in the holes.

People often ask whether we foresaw a PC on every desk. Certainly we did, since we knew Moore's law. I wrote a paper in 1972 that predicted exactly that. We even predicted today's Tablet PC; there's a picture of it, from the late sixties, in the text of Alan's talk on the Academy's web site. And we were pretty cocky, so we thought people would want the bit-map displays, laser printers, networking, what-you-see-is-what-you-get editors, drawing programs, file servers, and point-and-click e-mail that we built. One thing we didn't foresee was that software would become such an important industry; perhaps this was because it doesn't weigh anything.

People also often write about how Xerox didn't benefit from our work. Actually Xerox benefited a lot, in high end computer printing, where they've made billions of dollars. Xerox also brought a wonderful office system product to market in 1981, the Star. It was much better than anything we built in research; in fact, that was its problem, because it was too

expensive. It's ironic that the researchers tried, and failed, to get the product group to build something less wonderful, but much simpler and cheaper. So this was a failure of product planning and marketing, not a case of technology left on the shelf.

Today's PC is about 10,000 times as big and fast as an Alto; in fact, the MSN Direct wristwatch I'm wearing is bigger and faster than an Alto. But the PC doesn't do 10,000 times as much, or do it 10,000 times as fast, or even 100 × of either. Where did all the bytes and cycles go? They went into visual fidelity and elegance, integration, backward compatibility, bigger objects (whole books instead of memos), and most of all, time to market. At least today's PC does obey Alan's dictum. It's like Kleenex: use it once and throw it away. The Alto, much to his frustration, lasted for eight years.

What can we learn from the Alto about the future of computing? Well, in Alan's words, "the best way to predict the future is to invent it." I'm constantly amazed at the number of people who think that there's not much more to do with computers. Actually, the computer revolution has only just begun.

Looking at the history of computing from 50,000 feet, you can see that computers are good for three things: simulation, communication, and embodiment. We started with simulation, of nuclear weapons and payrolls. Twenty-five years later communication started to blossom, with e-mail, the Internet, and the web. Today, after another twenty-five years, we're in the earliest stages of embodiment: computers interacting with the physical world. The Mars rovers and the Roomba vacuum cleaner are just the beginning. For twenty years I've been predicting that the next decade would be the decade of household robots. Well, now we have one, so I was only twenty years too early.

And we're also in the early stages of computers that can understand speech and pictures, drive cars, and repair themselves. I dictated this talk to my computer, which is quite a bit faster than typing it, and much more comfortable. I've proposed a grand challenge research problem: reduce highway traffic deaths to zero. This can only be done by making the cars drive themselves, at least in emergencies. We have good enough cameras, microphones, brakes, and steering, so this is a pure computer science problem: vision, modeling the world and its uncertainty, planning, system reliability. And success would have valuable byproducts: existing roads could carry a lot more traffic, and drivers could spend their time on something else. Perhaps someone here can explain to me why I haven't been able to sell this idea.

The four of us being honored tonight did only a fraction of the work of building the Alto system. About 50 remarkably talented people worked on it for about eight years. Many of them went on to found major companies. I wish I could say something about each one of them, but it would be overwhelming to list even the major contributors.

My wife Lois is here tonight; we've been married for 35 years. She works in biology, which will be an even more exciting field than computing in the next few years, and she wrote her PhD thesis on the Alto. My son Michael is also a biologist, a postdoc at Rockefeller. He is here with his wife Min-Young Kim, a violinist who plays in the Daedalus Quartet. My other son David, a writer, is in Machu Picchu in Peru and couldn't be here. There's a bright future before all of them.

Thank you.