DOI:10.1145/2822519

Len Shustek

Interview

An Interview with Fred Brooks

ACM Fellow and A.M. Turing Award recipient Fred Brooks reflects on his career.

ACM FELLOW FREDERICK ("Fred") Brooks, recipient of the 1999 A.M. Turing Award, has made landmark contributions to computer architecture, operating systems, and software engineering. After earning a Ph.D. in Applied Mathematics and Computer Science from Harvard under the legendary Howard Aiken, he worked for IBM on several landmark computer systems, most notably the System/360 series that came to dominate mainframe computing for decades. He left IBM in 1964 to found the Computer Science Department at the University of North Carolina, from which he retired at the end of the Spring 2015 semester.

Noted software architect Grady Booch conducted an oral interview of Brooks in Cambridge, U.K., in September 2007. The complete transcript[a] of this interview is available in the Computer History Museum's oral history archive; presented here is a condensed and highly edited version designed to whet your appetite.

—*Len Shustek*

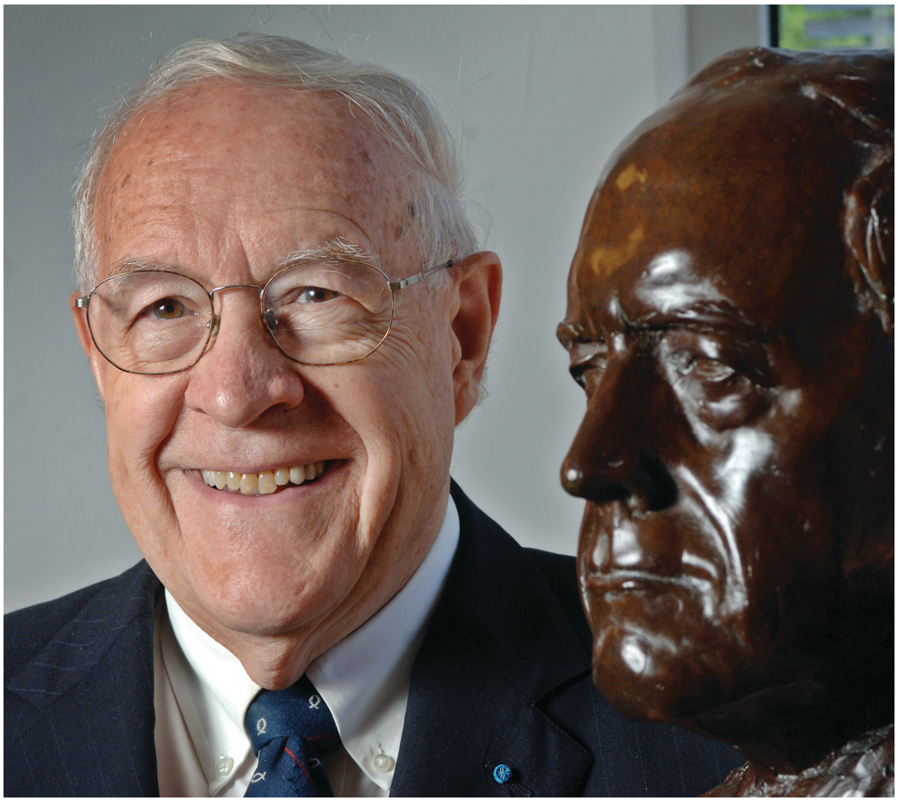
Fred Brooks with a statue created in his honor at the University of North Carolina Department of Computer Science.

PHOTO BY DAN SEARS, THE UNIVERSITY OF NORTH CAROLINA AT CHAPEL HILL

Falling in Love with Technology

I was born in Duke University Hospital because my father was teaching chemistry at the University of North Carolina, and that was the nearest hospital at the time. When I was about six months old he decided to change careers and go to medical school, so we settled in Greensboro, N.C. It was a great place to grow up because of the superb school system.

My favorite subject was physics, but I always had a fascination with business equipment. When the local corset factory went out of business, I bought filing cabinets for $4 apiece and a Burroughs comptometer that I think I paid $35 for. I made my own McBee card keysort system[b] for keeping track of my map collection. So I have always been fascinated with the kind of machines that process information. When I was 13, I read in a *Time* maga-

a http://www.computerhistory.org/collections/catalog/102658255

b An edge-notched card filing system invented in 1896; see http://en.wikipedia.org/wiki/Edge-notched_card zine in the little town library about the Harvard Mark I computer and I knew from then on that was what I wanted to do.

Studying Computers at Harvard

I did my undergraduate major at Duke University in physics, but I also studied math, economics, and accounting as well as the humanities because that was also my interest. The head of the physics department was determined that his best students should go to Harvard to study physics, and two of us did. When I told my advisor there that I really want to study computers, he said, "Fred, you're too late to get in on the ground floor, but you can catch the first landing." That meant I had a chance to meet and really get to know J. Presper Eckert, John Mauchly, and Konrad Zuse. Howard Aiken was my thesis advisor. So I got to really know the pioneers even though I was not in that generation, and that has been a great joy.

I went into the computation lab, a very small group that was part of a division of engineering in applied physics. There was Aiken, who was the boss, and two young instructors, Ken Iverson and Bob Minnick. Every day the boss was in town the whole crowd gathered for coffee at 5 P.M. in the machine room, which had the Harvard Mark I on one side and the Harvard Mark IV on the other side, each about 60 feet long. We would talk for a half-hour or 45 minutes, and then he would go home to supper and we would go back to work.

Programming the Harvard Mark IV

We programmed the Mark IV starting the first semester. My present colleague Bill Wright and I undertook, for our first year project, to write a program that would analyze melodies and create synthetic melodies using an eightfold Markovian process. We chose common meter hymns because we could find a lot of them that had the same metrical structure. We transposed them all to the same key, then we analyzed the transition probabilities of the melodies. It took us three years to get done, but we got music you could pass off on any choir.

The Mark IV was programmed in decimal absolute. It had 230 registers, 10,000 words of program storage, and 4,000 words of backup drum storage. There wasn't an assembler, but Aiken had designed a relay box that enabled you to encode it algebraically. Our first program for determining the Markov frequencies ran about 2,500 instructions. We were allowed two one-hour slots on the machine during the semester, so we did extensive desk checking. It is the only big program I ever wrote that ran right the first time.

When I was 13, I read about the Harvard Mark I computer and I knew from then on that was what I wanted to do.

Howard Aiken as an Advisor

At the end of my first year Aiken told Ken Iverson "I would like you to prepare a course on the application of computers to business." Nobody had ever taught any such course anywhere in the world. I had had trouble with a course in boundary value problems and was not continuing my NSF fellowship, so I went to Ken and said, "Can I be your teaching assistant, because that's right down my alley?" Out of that came our book *Automatic Data Processing*, which went through an edition based on the IBM 650 and then six years later an edition based on the 360. Ken was fully as important in my education as Aiken was.

For my dissertation, Aiken said "I want you to design a machine specialized for payrolls." That sounded like a good topic, even though he was more interested in the machine and I was more interested in the design methodology—how you get from the requirements to the machine.

His hypothesis was that by specializing a machine for payroll you could make significant improvements in cost performance. That turned out not to be the case. By specializing for serial file processing you could get an improvement, and that was essentially the machine I designed for my dissertation, but there were no further economies to be had by making it specialized for payroll.

Aiken was a very impressive person, and did something I cannot do with my students: he came to my office every day during the year I was working on my dissertation and wanted to see the prose that had appeared since the day before. And guess what? Some had appeared since the day before.

Helpful Summer Jobs

I was offered a summer job at Marathon Oil Company in Findlay, OH, using an IBM 650 to mechanize payroll. That was priceless experience. The complexities in business data processing are that you are trying to lump a wide variety of cases under a single process, so typically on the 650 when it was doing business applications, half of all instructions would be conditional branches. That is not true in scientific computing.

It helped me as a computer architect that I was also doing scientific computing. I spent a summer at North American Aviation doing missile tracking and database building. A summer at Bell Labs trying to identify which party on a four-party line was dialing this call. A summer at IBM in Endicott, NY, where I learned punched card machines and was in the physics department doing acoustics. Those four summers were a priceless part of my education, and they exposed me to radically different corporate cultures and radically different geographies.

Working on IBM's Supercomputer

Steve Dunwell, who was project manager for Stretch, came through Harvard in 1956 looking for people. His offer was very attractive: a chance to work on the world's fastest computer. I leapt at the opportunity. My wife Nancy, whom I had met the first Sunday night at Harvard, worked on the transistor circuits, and I worked with the architects under Werner Buchholtz.

IBM had been thinking about it for a year, and had just signed the contract with the National Security Agency to build Harvest, a variant of Stretch for breaking wire-wheel cryptographic codes. I was the only one of the architects proper who was native-born, so I could get the security clearance to work on it. Stretch, as the host, was about 15-feet long and 5-feet high and 5-feet deep. Harvest was a "plug-in card" that added another 20 feet of specialized electronics. It processed about 4 million bytes a second, and had about 250 bytes of instruction to set it up that would take half a day to write.

The purpose of Stretch was to make the fastest machine we could, cost no object. That is both liberating and tempting, but we did not succeed. Dunwell had set out to make a machine 100 times faster than the IBM 704. But the memories available were only six times faster, and the circuits were about 10 times faster. The idea was that by using more of everything we could get there, and you can't. We only got to 50 times faster.

It was declared a disaster and withdrawn from the market. But later IBM recognized Stretch technology enabled the 7090s, 7080s, and 7074s, and that the concepts became crucial for the System/360. Tom Watson, to his credit, went back and got Steve Dunwell out of disgrace and made a special award to him recognizing the important influence the Stretch had on the company's welfare.

Figuring Out IBM's Future

The Data Systems Division, which was the middle of the market in computers, recognized they had a product problem with their many existing lines. The vice president for engineering appointed a committee to study what a successor product line architecture might look like. I had moved to the research division, but I chaired that committee.

At the end of the summer I was asked to come back to the Data Systems Division as manager of architecture. We undertook to design a new product line called the 8000 series, built around Stretch concepts. There was a small scientific computer, small and mid-sized business machines, a grown-up version for high performance use by insurance companies and utilities, and a grown-up version for scientific computing. We worked very hard, and in January of 1961 we had cost estimates and market forecasts, plus a plan that involved creating new markets by making these machines communications oriented.

In January we did an all-day presentation of the 8000 series with the brass up from Armonk and White Plains. The whole program was very well received, except for one fellow sitting in the back who just got glummer as the day went on. That was Vin Learson, and that is not who you want to get glummer, because he was executive vice president of the company. Well, that night he fired my boss. He brought in Bob Evans and told him "If it's right, make it happen; if it's wrong, turn it around."

Bob spent three weeks looking into it, then took me out to dinner at a restaurant in Poughkeepsie and told me he had decided it was wrong. We were losing market share to the Seven Dwarfs.[c] Everybody was out after us. We were obsolete. We were fundamentally address-size limited. We could not attach more memory to the 7090 or the 7080, and the applications were hungry for more memory.

He proposed to wait for a new semi-integrated circuit technology that was going to be three years down the road. The problem is, how do you hold the market in the meantime? My plan would get out there now, although it had some fundamental difficulties.

He was right and I was wrong. He had two parts of his vision. One was that we ought not to do a new product line for the Data Systems Division; we ought to do a new product line for the IBM company, big and little. The other was that we ought to make the new product line coincide with the new technology.

We fought back and forth. We went to the Corporate Management Committee in March, and I won. That did not slow Bob down a bit. Bob is unstoppable. He went to the Corporate Management Committee in May, and he won. It was over. This meant stopping all the 8000 series projects in the Poughkeepsie lab and reassigning all the people. His plan was to do temporizing machines—the 7094, the 7080 model 3, and so forth—to hold the market as best we could until we could get there with the new product line using the new technology.

I had gone to a retreat up at Saratoga Springs to spend a week ironing out who is going to do what, and to make sure my boys landed on their feet in the reassignment. I was on the way out back to research when, to my utter amazement, Bob asked me to take the new product line. The crown jewels!

I was dubious. We had been fighting pretty hard. But Jerry Haddad, who was Bob's boss at the time, said, "I never knew anybody that regretted working for Bob Evans." So I gave it a try, and it went really well. We clicked, and we fought shoulder to shoulder, side by side, the rest of the way. Bob is one of four great bosses I had in my life.

Designing the System/360

Architecturally we started out in the data systems division pursuing a stack architecture, which is what Burroughs had done. It turns out that works great if you've got a transistor register for several levels, and it does not work great if you've got it all in memory, and you are having to pull and push all the time.

The problem is addressing. If you put addresses big enough on the littlest machine, your littlest machine is serial by byte and that means you spend a lot of cycles fetching address bytes that you are never going to use, and that compromises your performance. The stack architecture was a way of addressing that by not fetching as many addresses, but when we went through the performance and cost estimation in March 1962, the stack machine was working fine from the middle size up but was not competitive below because of all this pushing data in and out of memory. We had accepted the assignment of making the whole product line upward and downward compatible with one architecture.

So we scrapped it and had a design competition, which was Gene Amdahl's idea. I think there were 12 or 13 internal teams, mostly of three or four people. I said the decision of the judge will be final. Gene's team and Jerry's team came in with essentially the same concept, which was to use the base registers that Philco had introduced in a machine a year or so earlier. Base registers meant we could get by with short addresses.

There was one very big difference. Gene's machine was based on the existing 6-bit byte and multiples of that: 24-bit instructions and a 48-bit instruction or floating point. Jerry's

c Burroughs, Sperry Rand, Control Data Corporation, Honeywell, General Electric, RCA, and NCR.

machine was based on an 8-bit byte and 32-bit instructions, so 64-bit and 32-bit floating point. This is not a real happy choice. There are strong arguments each way, but you want your architecture to be consistent. You are not going to have an 8-bit byte and 48-bit instruction floating-point word.

It was our biggest internal fight. Gene and I each quit the company once that week, but Mannie Piore, the senior scientist in the company and a person of great wisdom, got us back together. I had made the decision for the 8-bit byte. Gene appealed to Bob; but Bob affirmed it. Of all my technical accomplishments, making the 8-bit byte decision is far and away the most important. The reason was that it opened up the lowercase alphabet. I saw language processing as being another new market area that we were not in, and could not get into very well as long we were doing 6-bit character sets.

When we announced the 360 in April 1964, I said to my team is "Tonight the lights will be on in the other guys' offices"—those of our Seven Dwarfs competitors—because the project had been kept pretty well secret. And the orders just started pouring in.

Operating Systems

In 1962 we formulated a plan to have four software levels, known as Romans I, II, III, IV. Because we had hardware compatibility across the line, the software levels had to be distinguished only by the memory size. W.B. McWhirter, who was the Data Systems Division president, took the software away from us, but they were busy, and they did not give the 360 software much attention.

Before the announcement I said to Bob, "Look, the machines are being released to the factory. The hardware part is done. The machines are on the track. Everything's rolling. But the software is in an utter mess. Between now and September, when I will leave to help found the Computer Science Department at the University of North Carolina at Chapel Hill, let me go over there and bail, and just see what can be done with the boat."

I changed teams, and we had a retreat off in the woods in February 1964. We came back with a totally different product plan that involved these memory levels, compatibility among them, a variety of language processors—two FORTRANs, a Report Program Generator, two levels of COBOL—and a modular operating system that would start at 16K. You have got to leave some space for the application programs, so we said the operating system has to be resident in 4KB. It turned out we could not do that, so we ended up making 32KB the minimum size machine for OS/360. We had a tape-based operating system, a card-based operating system, and a little disk operating system called Basic Operating System. Those got delivered on schedule in February 1965.

But the big operating system was in trouble. We patched and bailed, but by summer it was clear that we were still in trouble. Tom Watson invited me down to Armonk for one of these one-on-one luncheons in the executive dining room and said, "Why don't you stay here?" I said that I really wanted to get back closer to the technical level. "Well, will you stay another year and keep on bailing out the operating system? If you will, we'll send somebody to Chapel Hill to teach your courses. You go one week a month to get your department organized. And when you need a new computer in Chapel Hill, we will help." That was a good offer, so I said yes.

The workforce peaked at about 1,000. We did deliver the first version in April or May of 1965, but it was slow. It was November, really, before a respectable version was delivered.

The worst decision, which is documented in *The Mythical Man Month*,[d] was to take the software away from the architecture group and give it to the operating system manager. The architecture manager had said, "If you leave it with me, it'll be the same amount late, it will cost the same, but it'll be right." He was right and I was wrong, and that was a multimillion-dollar mistake.

Languages

Assembly language, surprisingly enough, posed a lot of technical problems. There were two entirely different schools of thought. On the commercial side, the assembly language served as a platform on which high priests in corporations wrote macros and the troops programmed in the macro language. In the scientific community, the scientists would write their own macros and they programmed in macro-enhanced assembly language. The question of how to resolve all this led me into the thickest technical thickets I got into.

The other set of technical decisions had to do with PL/I. One serious question is: Do we do independent evaluation of expressions so that you can factor out common sub-expressions? The answer was yes, but that leads you to peculiarities such as one divided by three equals zero because of data typing.

The basic concept was to make a universal programming language that would meld and displace FORTRAN and COBOL. ALGOL was not really a factor, although part of our product plan included delivering an ALGOL compiler for government reasons. ALGOL was popular in Europe and had a lot of important concepts in it, particularly having to do with subroutine

There is still an awful lot we do not know about how to build complicated systems and keep them under control.

calls and parameter passing, which we had to master and adopt. An important step forward in PL/I was the provision of compile-time mode, just like macro assembler.

If we had been smart, we would also have done a schedule-time mode of PL/I instead of doing JCL, the Job Control Language. But we weren't smart. The worst mistake we made was JCL. Its existence was a mistake. Building it on a card format was a mistake. Building it on assembly language was a mistake. Thinking of it as only six little control cards instead of a language was a mistake, because it had no proper subroutine facilities, no proper branching facilities. That we did not see it as a language was the fundamental problem; we saw it as a set of control cards.

Except in the U.K., IBM waffled about its support for PL/I. They did the same thing with APL—on again, off again—when a wholehearted support would have made it happen. Now, could you have displaced FORTRAN? No, I now think that would have been impossible. I think you could have displaced COBOL, whose community is more coherent. It is more top-down oriented, and PL/I is a better substitute for COBOL than it is for FORTRAN.

How Big the Project Was

My hardware development budget was about $100 million. That doesn't count all the I/O devices, the disks, the tapes, the new keypunches to accommodate 8-bit bytes and the larger character set, and the capital for the factories. The whole program cost billions. The software budget for the OS, not counting DOS, I think was somewhere in the neighborhood of $400 million in 1964 dollars, which would be $4 billion today roughly.

We recognized that building big systems is different from building programs. There is still an awful lot we do not know about how to build complicated systems and keep them under control. Tom Watson asked me a question at lunch: "You've managed the hardware program. You've managed the software program. What's the difference?" I said, "Well I can't answer that one but I'll think about it." That's where *The Mythical Man Month* came from. As I say in the book, managing software is more like managing hardware than most software people believe and it is less like managing hardware than most hardware managers believe.

On to Chapel Hill

I love to teach, and the atmosphere and the support at Chapel Hill in terms of my colleagues could not have been better.

One of the areas we picked for emphasis when we started the department was interactive three-dimensional computer graphics. I worked for 30 years with protein chemists, building tools to enable them to construct protein structures from crystallographic data. Then the virtual reality concepts came along as a natural extension of 3D interactive computer graphics. We have been doing that for years, looking mostly at serious applications as opposed to entertainment applications.

I would advise someone considering a career in the computer field to look at the intersection between the computer field and biology. The ablest young people today are often opting entirely to go totally into biology instead of computers, and biology offers the same promise today that the computer field did to me in 1953. On the other hand, the key to much biology is information: where it is hidden and how is it processed. For the computer scientist the interaction with biology and biologists is the golden opportunity. That is where the fun is going to be. C

d Frederick P. Brooks, Jr., *The Mythical Man-Month*. Addison-Wesley, 1975, 1982, 47.

Len Shustek (shustek@computerhistory.org) is the chairman of the Computer History Museum.