

## An Interview with Bob Rogers of IBM

If you are a regular attendee at mainframe conferences, you will have met Bob Rogers, or you will soon do so. It's unavoidable! Either way, you will not forget the high energy, effervescent spirit and non-stop monologue that both entertains and instructs - if you can keep up! This manic "envelope" contains one of the bright people who has helped IBM evolve its memory architectures, a Distinguished Engineer and software strategist. Bob is widely known for the informative talks he gives as a top-level "techie" for Big Blue. He's an easy pick for our Mainframe Hall of Fame.



**Bob Rogers**

### **Q.: When and how did you get into computers?**

**Bob:** I was in my second year of college in Manhattan College in NYC. College seemed a bit sophomoric to me, so in 1969 I switched schools to Marist College in Poughkeepsie, NY and got a job as an operator at IBM as I studied mathematics. I found programming and working with computers, even as an operator, to be so exciting - so absorbing. As John Ehrman says, I caught the first computer virus!

I was very fortunate to be an operator at this location, because this is where the operating system was developed. In those days, programmers would have to come to the machines for testing. I was lucky enough to observe the programmers, and let those with kind hearts to educate me a little bit.

I had access to the manual library in my building, and I taught myself FORTRAN from the Reference Guide! It wasn't until later that I found out that in the adjacent building IBM had a lovely technical library with books intended to teach people! I learned PL/I with a self-study course, and I was consumed with wanting to learn more about computer languages and make the computer do what I wanted it to do.

Sometimes, if a machine was standing idle, I would whip up a little program to make it do something cute.

One time I wrote a program to compute prime numbers. Now, most prime number-generating programs store the numbers that they already know to be prime. I stored them on tape, so that whenever the algorithm would go to the store of already-proven primes, it would make the tape move. This was like what you see on TV, where the tape moving is the "computer thing" in an episode.

I had a lot of fun being an operator on the way to becoming a programmer, and it didn't take me long to realize that this was probably more fun than mathematics.

It changed my life.

In another two years I graduated, and became a programmer at the IBM Development lab. I had two qualifications: I had been a successful operator for two years and I had a degree. As a result, I got to see more things than most college grads get to see. In my 38 years at IBM I have always drawn on my experiences as an operator and a programmer.

**Q.: You've done so many things at IBM. What do you consider your "short list" of achievements during your career?**

**Bob:** I've spent a disproportionate amount of time working on taking the whole mainframe environment and moving it forward to a new addressing architecture. It started with MVS/XA, when we went from 24-bit to 31-bit. It was during that project that I became (not to be immodest) an "MVS expert."

I was hired to work on the Nucleus Initialization Program. Basically, "NIP" is the system starting itself up from nothing to be a full-blown operating system. Since all of the major components of MVS need to initialize, and since I was the guy who had spent weeks studying the system initialization logic manuals and reading all the code, I was the guy who knew how initialization took place.

So, as the designers for the XA-level components (such as the XA Real Storage Manager, the XA Virtual Storage Manager, the XA I/O Supervisor and so forth) needed to know how to get their systems initialized, they would come to me. And I'd have them educate me, by asking, "What are you doing? What are you trying to accomplish?"

In this way I quickly got a broad view of the operating system. This came just at the right time for me. NIP is a component that is extremely "boring" 99 years out of a century, but I was there for the one year that things really changed.

I re-wrote the program that would get control when you do the load function (there's no load button any more). That is the first program to be loaded by the hardware - I wrote the XA version of that program.

This put me in a very good position to be one of the people who helped evolve the system toward MVS/ESA when that time came. That was interesting, because the things we learned in getting the OS to MVS/XA were inapplicable. So the combination of the two projects (I'm leading up to 64-bit here) made me uniquely qualified to lead that effort on the "straight and narrow" in an era where resources are tighter and the mainframe isn't "king." We had to be judicious about our use of resources and timing to be able to navigate our evolution into the 64-bit world in a successful way.

I think it's worked out pretty well in terms of z/OS delivering capabilities in a timely manner, not wasting time and energy and causing incompatibilities by doing things that didn't need to be done until they did need to be done.

So if there's a thread that runs through my entire programming career, it's moving to new storage addressing architectures. And by coincidence, it happens to be something that touches close to the hardware, close to the architecture. That's given me the opportunity to work with a lot of really smart engineers, CPU architects who help make the whole system work together.

**Q.: What can you tell me about where you see the industry going in the next five or ten years?**

**Bob:** I can't tell you a lot, because the five to ten year timeframe is not what I do. I'm much more "tactical". I do have a three-year window, because in three years we'll probably deliver what my colleagues are thinking about today.

Where is the industry going? There are rumors that Moore's Law has temporarily petered out (that everything can "double" every eighteen months). However, there is some word coming out of Research that makes people say, "We're going to be able to keep Moore's Law cranking". As far as whether the engineers are going to be able to continue to deliver compute power, I really don't know.

As more compute power becomes available, more powerful programming technologies can be used, by which I mean that programmers can get lazier and lazier and still get the job done. One of the biggest struggles we have now is to absorb these new programming technologies that consume compute power as if it's FREE and to try to be competitive with the other platforms.

Next week I'm giving a talk at the System z technical conference that will be entitled "System z Trends," which suggests that I'm going to talk about the future. But instead I'm going to talk about where we've been recently, which will seem like the future to our customers. That's because they're stolid and they move slowly, because they must move very sure-footedly. They don't absorb new technology just because it's "fun".

So, to me the recent past might be the future for other folks.

Linux is becoming more important up and down the platform line. That's one thing. It'll be interesting to see what really does happen with Moore's Law.

**Q.: Can you expand on Moore's Law?**

**Bob:** The gist of it is that "things double every eighteen months". Like CMOS processor speed: instruction processing speed would double every eighteen months. And we've been seeing that for the last decade. Memory sizes: We've been making memory smaller and smaller and so we see the same doubling in storage capacity.

There are a lot of people who have ONLY known the effects of Moore's Law. They think we're on a rocket ship; the sky's the limit. But an end has to come. Eventually you have to have a certain number of atoms or electrons in order to represent a "one". It can't go below one electron, and it probably can't even come close to that. So, adjustments will have to be made, as we had to when we got onto this CMOS rocket-sled.

**Q.: Who do you consider your mentors?**

**Bob:** I really have two mentors. One is Dave Stucki, who retired from IBM over a decade ago. Dave was one of the best debuggers and designers we had.

Because I was not a morning person, I tended to stay around late, and I'd run into Mr. Stucki. I wasn't afraid to ask him questions and he responded wonderfully to that. He'd explain things to me, and I would watch him work. It was from him that I learned, "If you are the guy to whom other people come to find out if their design is any good, then you get a free education!"

Once you get to the point where you're the acknowledged expert, people come to you to give you tutorials on their proposed design. Obviously, the amount of knowledge you get is going to increase. The reputation of being someone who has a lot of knowledge and is willing to help other people with their designs; it's the best way to learn.

My other mentor is still working with IBM. That's Bernie Pierce, a person who is very well known in the performance and capacity parts of the world - CMG kinds of stuff. Bernie, in my estimation, is the best performance designer for the MVS operating system on the planet! He was instrumental in the details of the workings of the dispatcher for the specialty engines: the zIIPs and the zAAPs we've come out with. These are the things Bernie attacks from the technical point of view. But the real reason I consider Bernie a mentor is I was able to make some sort of relationship between what we're doing with technology and the fact that IBM is here to make money. We need to do things that are profitable, not just because they're "cool." Bernie's great from a technical point of view, but the unique thing I've gotten from him is that performance has to do with business.

**Q.: How many hours a week do you work, and how do you spend them?**

**Bob:** Ever since I was an operator, it has been difficult for me to tell when I'm working and when I'm not. When I was a beginning programmer, I used to program at my kitchen table. My workday was spent talking to people and punching up code or testing it - things that I needed a computer to do. But I did my original coding at home - away from all the people that I could interact with at IBM.

For me, a very gregarious person (and garrulous as well), there's a huge temptation to get up and go talk to someone. We have so many interesting things going on, so many interesting people. So, all my own work was done off to the side as if it were a moonlighting job. I would spend my days working on other people's problems and getting to understand the system and so forth.

I don't think that anyone who's been involved with computers would be surprised at this phenomenon: You're just about to shoot the last bug, so why not stay another fifteen minutes? Then, you realize that you're falling asleep and it's 4:00 in the morning!

Nowadays, with the Internet, my job has changed considerably. I mean, if you're going to work with people all over the country, or all over the world, what's the advantage of being in your office? If I have a morning that's dominated by teleconferences, I'll probably stay home.

So, it's hard to tell what part of my day is working and what isn't, and I suspect that's true of a lot of people these days.

**Q.: If you had it to do all over again, what would you have done differently?**

**Bob:** I think we learn from our mistakes, and if I were to do it differently to avoid making some mistake that I actually made, then the lesson I learned might NOT have been learned, and I might have made a mistake with worse consequences.

So far, I'm doing pretty well. I've been working at IBM a long time, I'm being interviewed for NaSPA/*Technical Support* magazine. I'm a Distinguished Engineer at IBM. I think I've done "OK" for a hyperactive guy. Well, formerly hyperactive. Now I'm too OLD to be hyperactive. Now, I'm just fidgety!

**Q.: What's your legacy? What will you leave behind?**

**Bob:** People keep asking me whether I have someone I'm training to replace me. I would love to do that if I could find someone whose temperament - whose nature - was similar enough to mine. But that I can't find, so I have a number of people who I mentor.

I once gave a talk to some interns at IBM and at the end of the talk I said, "If anyone is interested and has more questions, please contact me." And this guy did. I don't think that it was because of anything I said, but perhaps because he already had that kind of desire in his head. When we met, it crystallized. He would like to emulate me, which will be interesting because of our different temperaments and skills. He's coming along.

It turns out to be more challenging for me, because he's hearing impaired and I don't know American Sign Language. So, our communication needs to be very structured because when we schedule our bi-weekly meetings we have to have an interpreter. Anyone who knows me knows I like to be verbal and feel I can communicate fluidly when I speak. I like talking a lot, not typing a lot. So, for me, it's personally a challenge to be forced to communicate in such a structured manner. But, it is more than worth the inconvenience because he has so much potential. He and other young people like him will have to carry the flame in the more distant future.

Another person that I mentor is a young woman who has worked with us for some time and I have worked with for a few years. She is just so hard working and bright and extremely committed to excellence in all that she does. I won't be mentoring her for long at the rate she's going. In the near term, I see people like her keeping the flame burning that was lit by my predecessors - keeping the "mainframe" ahead of the pack and running the real business of the world.

**Q.: What would you suggest to new people coming into mainframes? What should they learn? What should they be interested in?**

**Bob:** The mainframe has two personalities, really. It has the legacy personality that I say runs western civilization (despite what others may say). A Martian looking at Planet Earth would probably report to his superiors that meaningful computing on this planet is, to a first-order of approximation, COBOL on CICS. And that's true for the last twelve years that the mainframe has been dead. But yet, in order to grow, and because customers are migrating or evolving we do need to address the new world as well (with JAVA, with XML, with [WebSphere](#)) in order to run the ERPs and CRMs of the world.

So my job and the job of the people I work with is to deliver real value and to show that we're moving forward. Perhaps not as fast as other platforms, but that's because we have to be really careful. We run the really important stuff!

I think it's important to know yourself, and find out how you want to grow personally. If you're a technologist, then that path will open up for you. If you want to help the business grow, then that path will become yours. If you want to advance yourself no matter what, then that will be your road. It truly does take all kinds of people to run a successful business. But know yourself first.

*NaSPA member Robert Shimizu provides first-level technical support for Cole Software.*