

The Mainframe Hall of Fame: An Interview with Ron Higgin, formerly MAINVIEW CTO for BMC Software

By Robert Shimizu



Ron Higgin

ONE CANNOT ATTEND SHARE or an IBM Disclosure meeting without encountering the tall and genial Ron Higgin. Ron's affable exterior conceals one of the brightest minds in mainframe software today. He is a perennial contributor to the industry at nearly every level,

with a career trajectory that we all could be proud of. By pure merit, Ron has become a strategic architect, someone who looks years ahead for companies like BMC and the former Boole and Babbage.

Ron kindly allowed me to interview him for this, the first in a series entitled, "The Mainframe Hall of Fame."

Q: *How did you get into computing?*

Ron: I was in high school in Brantford, Ontario Canada when the University of Waterloo hosted a "Computer Science Day" program for the better math students. I attended, and got hooked! I returned the next week and muscled in again. Soon I ordered the IBM manuals and began to read them.

I entered the University of Waterloo with the idea of becoming a nuclear physicist. However, I spent more and more time in the computer center. I joined the Computer Club and learned Assembler language and PL/1. I learned a great deal about coding by reading IBM's OS/360 source code. At the end of my first year I knew some OS/360 internals, Assembler language and PL/1. It was 1967-68.

Halfway through my second year in university, Mutual Life of Canada hired me to work on a PL/1 project. I quickly became bored, so I looked for something else to do. I wrote a utility program to analyze and optimize the structure of multi-region overlay load modules. Next, I wrote a utility that read the linkage-editor generated load module control records (ERs, LDs, etc.), sorted them nine ways to Sunday, and generated the Linkage Editor statements required to build the optimal region overlay structure for that module. Finally, I built a dynamic loader that would pre-load certain segments of an overlay structure based on the daily input transactions.

In those days you couldn't include PL/1 DECLAREs from a common library. You had to include them in every source program before you ran the compiler, so I modified the compiler's input phase to implement an integrated in-line INCLUDE capability. IBM later looked at that and incorporated the concept into their PL/1 Checkout and Optimizer compiler products. That got me noticed.

One day I saved the company eleven hours of processing time by fixing a production problem (through direct storage alteration) at the Operator Console. One hour later, the Director of Systems Programming offered me a job. Within three years I was running the Systems Programming Department, and later the Operations Department.

Soon I realized that good communication was critical in achieving my department's objectives. So, I began a program of education. We offered regular classes for our Operations, Applications programmers and Management groups, so that everyone understood what we were doing. That's when I began focusing on my communication skills—learning how to teach.

One of the things I'm most proud of is my ability to develop people with raw talent. At Mutual I drew from the Operations staff because they already had a basic understanding of what was going on. Each new systems programmer was teamed with an experienced person. The "trainer" of the pair was concurrently being groomed to take over another's job—and so on. No one was promoted to the next level until they had trained their replacement. This structured approach to training and career path management built a great team, where at least two people could perform any given task.

Eventually I had twenty programmers, each trained to back up another person. Everyone knew what their next job would be, and people in Operations competed to be the next Junior Systems Programmer. It was one of the best teams in the industry. We didn't hire good people. We developed them.

We need more mentoring and career development in the industry today.

During this period I also designed and built deep systems software. I re-wrote a portion of the OS/360 MVT storage manager to implement intelligent hierarchy support. I developed a "User PROCLIB" facility that was later adopted and extended by Amdahl in their corporate data center. I designed an "ENQ Manager" subsystem that resolved ENQ conflicts (sort of a JES3-like function) that would otherwise have suspended the initiators, wasting system resources. I designed an early tape-management system. Ultimately, I designed and wrote a great deal of the code for something I called the VTAM Application Monitor (VAM); a sort of multi-user address space capable of running SPF as an application outside of the TSO environment.

When people from Boole and Babbage saw the response time our SPF users were getting they said, "That's fast! What makes it so fast?" "Well," I replied, "It's not running under TSO." They loved it! Soon I began to work for them part-time, which led to a full-time job. I was 33 years old and happiest when I was designing and writing code.

In late 1972 I became involved with the GUIDE (IBM) user group. During my 27-year association with GUIDE I held numerous positions, eventually serving as its President. I learned that if you can manage a group of volunteers, then you can manage anyone! You have to do it by merit and reason rather than by bludgeoning people.

I began as Director of Interactive Systems at Boole and Babbage, but over the years I worked on pretty much all of their products. In the early 1990s I became a Boole and Babbage Fellow, working on various strategic projects. In 1999 Boole was acquired by BMC Software and I became a strategic planner for them.

Q: *What qualities make a good programmer?*

Ron: Curiosity: Don't be satisfied that something works. Ask the next question: "WHY does it work?" Then, "How can I make it better?"

Don't be satisfied that something works. Ask the next question: "WHY does it work?"

Tenacity: Never be satisfied. Always ask the hard questions. When coding, think, "What will happen if the code breaks at this point? What if the power fails here?" Do a great deal of "what-if" thinking as you code.

This is why mainframe software is so reliable—mainframe programmers have always asked the hard questions. You don't see this on other platforms, especially Windows.

Never give up looking for the simple solution to a complex problem. If your solution is overly complex, then you don't have the right solution yet!

Q: *Do you think that holding lots of variables in one's head at the same time contributes to success as a programmer?*

Ron: There seem to be two ways: Some people use a "Memory Palace" to hold disparate elements. Their solutions seem "intuitive" to others because of their prodigious memory. Others are process-oriented: First they check one thing, then the next, and so on. They eventually get to the solution, but perhaps not as fast as the intuitive guy.

Keep the big picture in mind. If you don't know where you're going, you'll be surprised when you get there! Visualize the ultimate objective, and ask yourself, "How does this piece fit into the whole?"

Q: *Would the love of symmetry be a good tool for a programmer?*

Ron: No. Pattern recognition is important. An old acquaintance of mine—Bill Robb—taught me something I've never forgotten: "If you don't know what it looks like when it's working, you have no hope of figuring out why it's not working, when it isn't."

For example, capture operating system and communication subsystem traces when they're working. Then, when things fail you'll be able

to superimpose the failure picture on top of the working picture. You'll see what's different, and that will lead you to the source of the problem. I've used that concept ever since.

Q: *What is your greatest achievement?*

Ron: It depends on whether you mean at the technical or non-technical level.

On the non-technical level, it has been recognizing raw talent in others and developing it. There's nothing more rewarding than seeing someone who you've mentored become incredibly successful!

My favorite technical achievement was the development of a runtime environment in support of deep systems development. That's foundation of much of BMC's Mainview product line.

I wanted developers to be able to write product code in C or Assembler language and run C code in states and memory keys that were heretofore restricted to assembly language programs. Everybody said it couldn't be done; that you couldn't write C language programs that ran in SRB mode, or ran physically disabled, or things like that. I just looked at it as another problem to be solved.

SAS Institute gave me information I needed to create my own runtime environment: one that supported running C-compiled code in SRB mode, physically disabled, cross-memory mode, space switches, and so on. I built a whole infrastructure, one that allowed for total interoperability between C language and Assembler language and provided a complete abstraction of all the important basic operating system services. Now our programmers never had to worry about doing GET-MAINS or ATTACHs or ESTAEs or SETFRRs. A programmer could request resources in a very general way, and I would figure out under the covers how to deliver those resources.

I wanted to be able to run this whole infrastructure in either test mode or production mode, so that the developers could "flip a switch" and be able to use a debugger (Cole Software's XDC product) for diagnostic purposes.

Programmers often forget to release storage, and end up with orphaned storage, especially common storage. So, I also took care of recovery and cleanup operations. Those kinds of problems disappeared, significantly improving the reliability of all our products. That was the BIG accomplishment I can point to.

Q: *Do you think mainframes will be here in five or ten years?*

Ron: Yes, I do. The trend toward faster machines in smaller footprints will continue, but not as rapidly. Until now, we've gotten throughput gains by shrinking cycle time. We're pushing the envelope there.

We'll see near 1000 MIPS engines in a couple of years. After that, we'll need something different. My best guess is some sort of internal grid technology in the hardware, a networking of multiple discrete processors yielding a logical CPU with higher capacity, rather than doing it through physics. That is why IBM is working to improve the speed of interconnects between components.

Profit margins on mainframe hardware will be pressured, continuing the downward trend in cost.

The big systems that run the world (like the financial systems) are going to stay on mainframes. These companies aren't going to re-write their systems to run on platforms that haven't been time-tested. They simply can't afford to mess around with "bet-your-business" applications and data.

Smaller guys are a different story, but not the big businesses. Some legacy trading companies are experimenting with hosting selected

functions on non-mainframe platforms with some success, but not for their core technologies. The big government agencies (the ones you can't talk about) are not coming off the mainframe—no way.

I think we'll see an accelerated movement towards true "platform-agnostic" plug-and-play capability, where you will be able to take systems management products from a BMC, or a CA or an IBM and plug them into a Service Oriented Architecture-based management system.

The mainframe is an important player in the big enterprise, but it's not the only platform. It's GOT to be integrated end-to-end with other platforms. The only way to do that is by implementing standards that facilitate plug-and-play, not just for systems management products but also for the pieces that make up what we think of as a traditional application. It's got to happen.

In five years, we'll see some of those systems deployed. In ten years they will be pervasive.

Looking at the applications layer, I think we'll see SOAs mature and become a viable methodology. Today, despite the hype, they're missing a lot. In particular they don't have the reliability characteristics of applications hosted by CICS, IMS, DB2, etc. Management for SOA based applications is in its infancy now.

SOA-based applications are really *virtual applications*. Instead of being one monolithic piece of code, they are discrete services integrated by an "orchestration framework." In essence this framework forms what we have traditionally thought of as an application.

Today, the technology to manage SOAs isn't there yet. But it's off and running.

Q: *What advice would you offer to people starting out on mainframes today?*

Ron: I would say, "Keep your perspective, and have an open mind." There are great ideas that emanate from the mainframe world, but the mainframe world doesn't have a monopoly on great ideas.

One might ask, "What is the best programming language?" The only correct answer is, "It depends on the application." Same with hardware—it depends. If the data, is key to the application and it's "bet-your-business" data, then I want that data on a mainframe where I have some faith that it's backed up and is recoverable.

Now, if the data is going to be on that platform, then where is the best place for the applications that access that data? There's only one answer to that question: As close to the data as possible, or you'll have latency problems. If the applications can't be hosted by the same z/OS system that owns the data then the next best place is in another partition on the same physical box running another z/OS system—or something like Linux.

If we're talking about a web-server, or not "bet-your-business" data or data that has been traditionally stored off the mainframe, then I might have a different set of answers to those questions.

One thing never changes: The application code that operates on the data must sit next to the data. Anything else will lead to performance issues, or (worse) availability issues.

So, keep an open mind. Use the right mix of hardware and software to achieve the desired result.

Another observation: those getting into the computer business must learn the value of discipline, and leverage it to their advantage.


Suppose you write a piece of code and ask, "What will happen if a failure occurs at this point?" An undisciplined programmer might answer: "The probability of that happening is so low I'm not going to worry about it."

Another (usually wrong) answer is "I'll just duplicate the data somewhere else." That leads to problems too. Newcomers need to ask the hard questions. If they do that, they may stumble upon innovative solutions to complex problems, AND they will gain a competitive advantage in the job market.

Discipline is why mainframes still exist. If the discipline in the mainframe industry was present on other platforms, mainframes might not have survived. Why must we re-boot Windows systems with annoying regularity? We certainly don't accept that in the mainframe world!

Q: *What would you have done differently, if you had your career to do over?*

Ron: I wouldn't change a darned thing. My time at Mutual Life gave me my grounding in software design and development, operations, communications, teaching and management. Later I was able to bridge the gap between developers and customers. I learned the value of having a solid architecture in place prior to building complex systems. That experience led me to take on higher-level architectural responsibilities. By the time I left BMC I was Chief Technical Officer for most of the Mainview product line. I spent a great deal of my time thinking about how to deliver truly integrated end-to-end management solutions that would deliver real value in today's open multi-platform world. Each phase of my career has served as a stepping stone to the next.

Nothing I've done has been a waste of time. I've made literally hundreds of friends along the way—people who have helped shape my career, people who I respect and people who I continue to learn from. It has been a joy and a privilege to have had the opportunity to know and interact with these people over a 36-year career. It's been quite a ride! 

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