TechChannel

Unlocking COBOL Business Value

→ How the COBOL Working Group is changing perspectives while closing the skills gap

→ Reg Harbeck on leveraging modern COBOL features to maximize cost savings

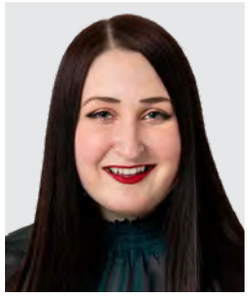
SPONSORED BY

software AG

MICRO FOCUS®

contents

Reversing COBOL Misconceptions



The ubiquity of both the IBM Z platform and COBOL programming language is at least partially due to the forethought IBM put into each in 1964 when the first mainframe was launched. COBOL was optimized for business processing with no need to recompile programs. As a result, it continues to run the world's economy today.

But many COBOL shops have fallen into a pattern of ignoring new programs or applications. To unlock and maximize business value—including cost savings, performance improvements and more—it's important to maintain existing code while leveraging the latest COBOL enhancements.

During the pandemic, COBOL faced a difficult misconception. With heightened unemployment rates in New Jersey, many couldn't file an online claim. The COBOL-built back-end application underpinning the state's unemployment system was blamed. In reality, challenges came from overloaded servers—but this led to other misconceptions surrounding a perceived lack of COBOL skills. Busting these myths is largely why the Open Mainframe Project COBOL Working Group emerged.

> **"Those of us in the mainframe ecosystem know the importance of advocating for the IBM Z platform, COBOL and other staples that keep the world running."**

Those of us in the mainframe ecosystem know the importance of advocating for the IBM Z platform, COBOL and other staples that keep the world running. When we come across misconceptions, we work to reverse them—and that's a collective community effort.

Keelia Estrada Moeller, Senior Editor

Bringing COBOL Front and Center

The Open Mainframe Project COBOL Working Group works to increase awareness and provide education

BY JIM UTSLER

All it took for COBOL to make the mainstream headlines was a pandemic. When COVID-19 began resulting in mass layoffs, many unemployed people in New Jersey found that filing an online claim was nearly impossible.

Many **New Jersey authorities** blamed the issues on the COBOL-built back-end application that underpins the state's unemployment system. More likely, the challenges were the result of overloaded servers or unanticipated pressures on other IT infrastructure components. This news then prompted other COBOL-related doomsday scenarios, especially involving the notion that COBOL skills are hard to find and develop.

This is in part why the **Open Mainframe Project COBOL Working Group** was established. The group aims to promote the programming language by changing its perception, including explaining why it remains so popular and what it can do. Many already know this, of course, including those in government and the financial, insurance, automotive, logistics and retail industries, among others, but it never hurts to both reinforce the message for them and introduce COBOL to others.

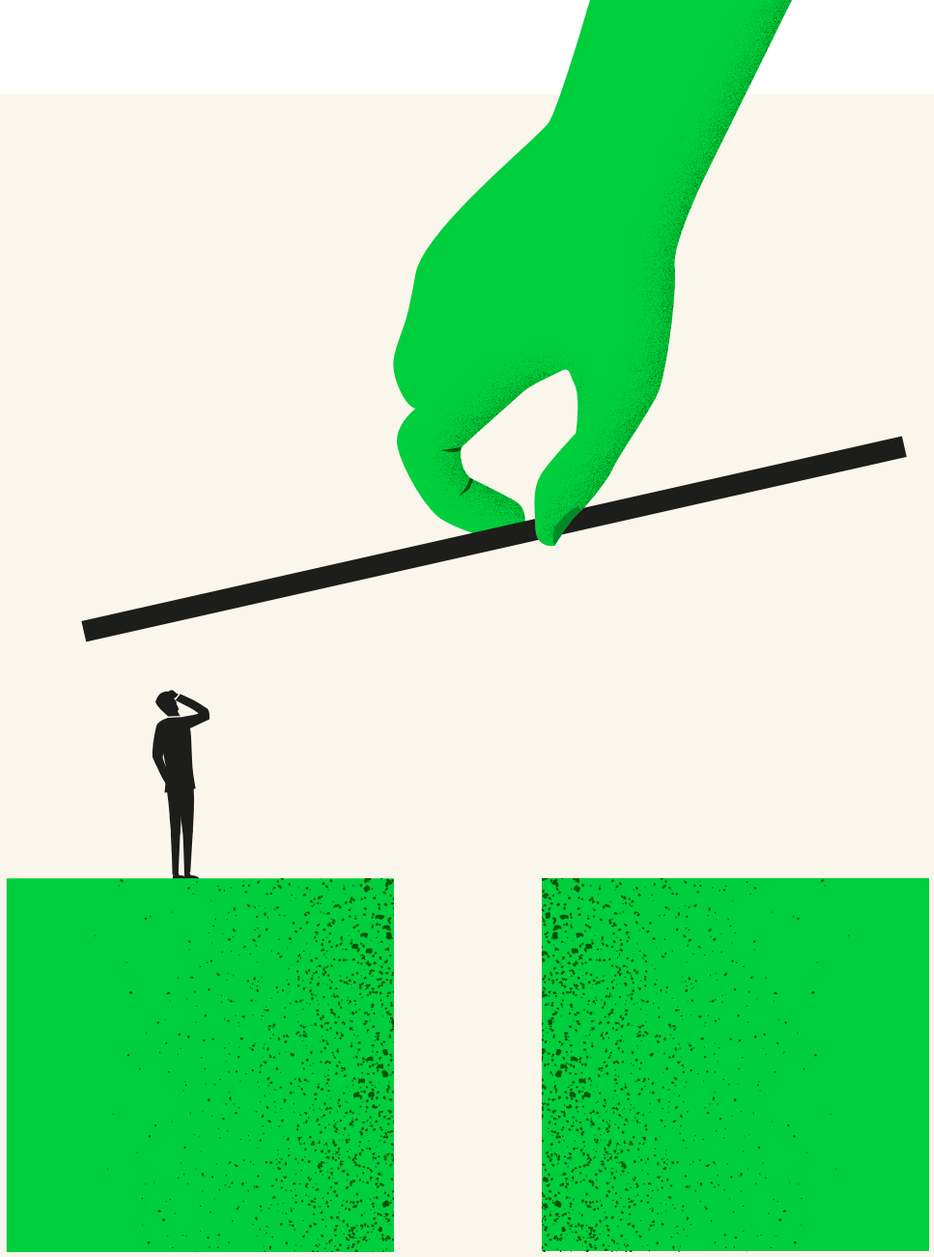
And this has become increasingly important, notes Derek Britton, global head of product marketing, AMC and IM&G Product Groups, Micro Focus. “A very substantial proportion of the Western world economy runs through a COBOL back end,” he says. This assertion is supported by the huge amount of lines of COBOL code that are still in use, with numbers ranging from 200 billion to 250 billion, depending on the source.

And as far as the availability of COBOL skills? “The short answer, and the best answer,” says Cameron Seay, adjunct professor at East Carolina University, “is that there’s currently no shortage of COBOL talent. What there is, is a disconnect between where the people who need the talent are located and where that talent’s located. They could be continents apart.” Both Seay and Britton are chairs of the Open Mainframe Project COBOL Working Group.

The pandemic essentially presented an opportunity to bridge the gap Seay refers to. When people began working remotely, geography was no longer an impediment to finding work, and many COBOL-savvy programmers suddenly found opportunities to share their expertise with any number of organizations, regardless of geographic location.

COBOL’S Built-in Flexibility

This begs the larger issue, as both Britton and Seay agree, about the lack of constant COBOL educational opportunities. Most colleges and universities simply don’t offer this at all, despite a



The pandemic presented an opportunity to bridge the gap between where COBOL talent is located and where that talent is needed.

“We needed to get the truth out there about COBOL. We needed to be clear about this for the market and for the people who report on it, so they could understand it. This is how the concept of the COBOL Working Group, as part of the Open Mainframe Project, came about.”

—Derek Britton, global head of product marketing, AMC and IM&G Product Groups, Micro Focus

desire from students to learn COBOL and a distinct need for COBOL programmers on the part of businesses.

Rather, schools are focusing on what they believe to be new languages and programming techniques they think will prepare their students for the future of computing. And getting COBOL on curriculums often requires large organizations and vendors partnering with local schools to both educate the next generation of COBOL programmers and help meet their more immediate requirements.

This may be the result of a potentially justifiable economic decision, not wanting to carry courses they’re afraid won’t attract students to their schools and programs, but it’s also somewhat shortsighted. No, COBOL doesn’t necessarily have the flash of, say, JavaScript, but it’s nonetheless critical to everyday operations for many organizations, even as they embrace the digital age. Indeed, COBOL, which is often updated to represent both today’s and tomorrow’s computing expectations, can comfortably coexist with other “shinier” programming techniques.

Without that type of application-to-application connectivity, banks, for example, may never have introduced mobile banking apps, thinking they wouldn’t be able to leverage their back-end COBOL business logic and present it in a user-friendly format. In fact, that mindset has led more than a few organizations down the tortuous, and frequently unsuccessful, path of ripping and replacing, getting rid of their tried-and-true COBOL applications in favor of third-party applications or those written from the ground up in another language.

“The decision to stop using COBOL represents a fairly seismic shift for most organizations of whatever size,” Britton notes. “You don't just walk away from COBOL overnight, not easily and not without significant potential impact. There are substantial downsides to this, including costs, time and effort that could’ve been better used on other projects that actually advance your business and make you more competitive. All the while, the very same COBOL is perfectly capable as a platform for digital innovation.”

COBOL is hardly the stodgy language some think it is. If somebody wants to build microservices in COBOL, they can. If they want to be build web services, they can. If they want to build containerized programs, they can. If they want to use mainframe resources such as CICS, Db2 and IMS, they can. COBOL can do all of this because it's constantly evolving to support the many resources that surround it. That's why Seay encourages his students to look at COBOL as yet another flexible, connected tool they can add to their toolbox—and they're taking him up on this.

According to Seay, "At most of the schools where I've taught, there are a lot of hungry students interested in COBOL. I'm also running a bootcamp this semester, for

“At most of the schools where I've taught, there are a lot of hungry students interested in COBOL. COBOL, which is actually pretty easy to learn, is very attractive to those who want to have a good career and make good money. I'm at a loss as to why more schools don't take advantage of this space.”

—Cameron Seay, adjunct professor, East Carolina University

which we had somewhere around 70 applicants for 30 slots. A lot of these are people who already have degrees in computer science or information systems but don't have jobs because schools aren’t equipping them with the skills the industry needs,” he says. “So, COBOL, which is actually pretty easy to learn, is very attractive to those who want to have a good career and make good money. I'm at a loss as to why more schools don't take advantage of this space.”

Speaking the Truth About COBOL

In part prompted by the negative headlines regarding COBOL and New Jersey’s unemployment system, which Britton says, “wasn’t necessarily correctly reported,” Seay, Britton and a few others decided somebody had to help set the record straight and evangelize for COBOL. Hence the creation of the **Open Mainframe Project** COBOL Working Group.

“We needed to get the truth out there about COBOL,” Britton recalls. “We needed to be clear about this for the market and for the people who report on it, so they could understand it. This is how the concept of the COBOL Working Group, as part of the Open Mainframe Project, came about.”

With the support of the people behind the project, they set up the Working Group to effectually act as a lobbying platform for COBOL. After all, if the mainframe community isn’t talking about COBOL, who else is going to? It should, as Britton remarks, “be on everyone's lips.” So, the group’s goal was and is starting discussions about why, for example, COBOL should be the mainframe’s default language and its many different use cases on the mainframe.

These ideas further spawned the **COBOL Training Course Project**, the goal of which, according to its website, “is to educates those developers or students who would like to learn COBOL skills with Microsoft’s Visual Studio Code

editor (VS Code) and extensions. These materials provide an overview of the language and real-life Enterprise COBOL demos to work on."

The COBOL Training Course Project focuses on COBOL from a technological perspective (much like Seay teaches his courses at East Carolina University) to help mainframers or business programmers of the future get the information they need now to move forward with the language in productive ways.

"I really want my students to understand what's going on under the hood," Seay says. "What almost all of them say is that, initially, over the first couple of weeks, they were terribly intimidated by the interface, but once they get past that, we take a step-by-step look at the actual functionality of COBOL. We address everything in digestible pieces, and that's quite successful."

Full COBOL Classes

Although the New Jersey unemployment-system misadventure may have initially besmirched COBOL's name, it's now being quickly rejuvenated by Seay, Britton and others who are now part of the recently created Open Mainframe Project COBOL Working Group, as well as the subsequent COBOL Training Course Project. The group currently has around 100 members and encourages others to join. By initiating discussions about COBOL, they're essentially giving it the recognition it deserves for being an incredibly sound and contemporary development technology.

And there's a need for this, according to Seay. "My classes are always full. They're full again this semester. There are over 100 students a year in the fall and spring semesters, and if those numbers were replicated at just a few schools, the perceived problem of a lack of COBOL programmers would just go away, and COBOL shops would all be that much better for it."

Additional COBOL Resources:

- **Open Mainframe Project Q&A technical forum and volunteer resource**
- **Open Mainframe Project COBOL Training Course on-demand webinar video**
- **COBOL presentation by Derek Britton from Open Mainframe Summit**

Optimizing Business Processing With COBOL and IBM Z

Taking advantage of modern COBOL features can help you realize strategic cost savings

BY REG HARBECK

"If it ain't broke, don't fix it," was probably not first said about COBOL or the IBM mainframe, but it might as well have been. In fact, individually they are clear embodiments of this phrase in their enablement of legacy applications that can run decades without needing modification.

COBOL, created in 1959 to 1960 with the leadership of visionaries such as Rear Admiral **Dr. Grace Hopper**, was designed to inherit all of the lessons of business computing and compiler technology and related matters learned during the formative years of electronic computing. It worked so well that it still runs the world economy today, with around betwen 200 to 250 billion lines of code, depending on the source.

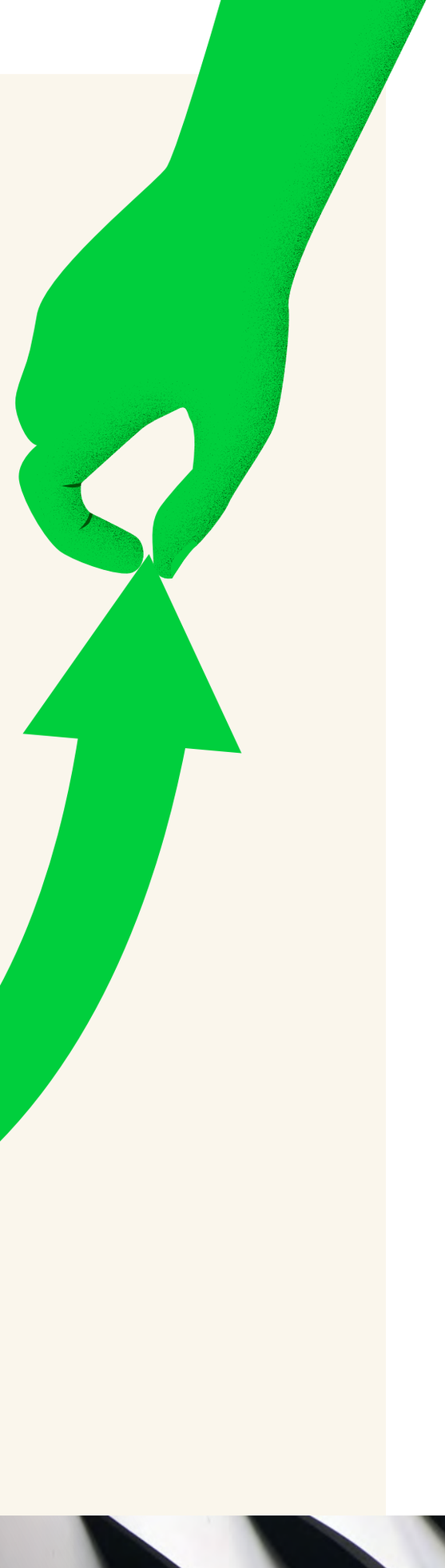
One of COBOL's strengths is that it was designed to make a programmer strategically think through how they would use their data, and then write programs with such English-like syntax that you can practically read the PROCEDURE DIVISION out loud to figure out what a given COBOL program does. Consequently, once a program had been written, it often didn't need to be rewritten until additional or replacement functionality was needed.

COBOL Has No Need for Recompiling

Of course, as long as hardware platforms kept changing their architectures, it was still necessary to recompile COBOL programs whenever new hardware came along. That changed on April 7, 1964, when IBM announced the System/360, promising that code that ran on any of them would run on all of them. IBM has kept that promise as new mainframes have been introduced. Consequently, a program that was compiled to run on one of the original S/360 computers may still run on a modern IBM Z computer—and there are likely some of them out there doing just that.

It was no longer necessary to change, or even recompile, programs because the design of both the language and the mainframe architecture were optimized for business processing—and for each other. So, a COBOL program that embodied sound business logic that was compiled for the IBM mainframe was in a sweet spot for the ages, which meant that future programs could be written using the time that had previously been invested in rewriting and/or recompiling the same old programs over and over again.

Enhancements available through the latest versions of COBOL, the compiler, the mainframe hardware and other features and interfaces have opened up an unlimited horizon of opportunity.



COBOL Keeps the World Running

Vast numbers of COBOL programs written for the IBM mainframe since the 1960s kept running, as reliably as the sun rises and sets. And if some of them did get sunset over the years due to obsolescence of functionality or replatforming, many just kept on ticking, invisibly keeping the world economy running.

Meanwhile, the world continued on, with every different kind of change, and if the functionality of these foundational programs didn't change, the world changed around them. Because of their inertia, the programs and programmers began to slowly diverge from opportunities to do new and better things.

Today, many mainframe COBOL shops have settled into a comfortable rut of only making strategic changes demanded by corporate initiatives, while ignoring any program or application that isn't in a current area of focus. We've all gotten used to not rocking the boat or suggesting arbitrary changes when we don't have enough staff just to keep making the mandated changes.

But here's the thing: These programs and applications aren't fossils. The operative assumptions about the business context that led to their creation have inevitably changed. And, the enhancements available through the latest versions of COBOL, the compiler, the mainframe hardware, and other features and interfaces, have opened up an unlimited horizon of opportunity for a new generation of explorers to uncover and unleash. That's important because it can trigger a financial windfall in an era when everyone's budget is so tightly wound that it's begun to cut off circulation.

Leveraging New COBOL Features and Functionalities

Of course, many different degrees of finding business value are available, and some qualify more as "low-hanging fruit" than others. The big one everyone seems to be talking about today is just recompiling your old COBOL with the **newest compiler** with optimization turned up to 11 (OK, 2). The new hardware features that the compiler can take advantage of, along with the optimization technology itself, can speed up your old code by an order of magnitude, with no change in functionality.

But that's just the beginning. The next degree is getting to know new functionality that has been added to COBOL since your programs were written that would improve the way they process their workloads. What you're really doing here is moving beyond just optimizing your code to beginning to optimize your people. Bring in the junior programmers and the visionary experienced ones and

have them study the features that have been added to COBOL over the past few decades. Then, glance over some of your legacy code and start brainstorming how you might enhance and optimize it, beginning with eliminating complicated work-arounds for old limitations that no longer exist.

Some of the newer features that can save steps include:

- Removal of memory constraints
- Automatic initialization of memory when allocating it
- Variables whose size can dynamically change when allocated
- Enhanced in-memory sorting features

Once you start to discover the value of updating your neglected COBOL code, you will give your

programmers a rewarding learning experience. You'll also enable your enterprise to rise to a higher level of functionality as ideas for bringing functions, features, data and business value will generate savings and profit in an upward cycle that will continue as long as your people are willing to pursue it. A good example of this is the JSON-enablement that allows COBOL to talk smoothly to applications all over the enterprise—including those running in Linux containers in Kubernetes under z/OS.

This can all lead to cost savings and profit opportunities. It can also enable career development and training, allowing you to encourage business sense as your best people suggest relevant improvements to bring your legacy code forward into alignment with today's prospects. Refreshing your corporate IT culture—particularly the somewhat resistant COBOL and legacy areas—with new insights and inspiration will help build a brilliant future with the most brilliant of platforms and languages.

REG HARBECK is a mainframe enthusiast who has been working in IT and mainframes for over three decades. During that time, he has worked with OSes, networks, computing security, middleware, applications and platforms across the industry. Reg is the chief strategist at Mainframe Analytics ltd.

Learn More About COBOL

- How COBOL REDEFINES the Way You Think About Business Programming
- A Look Back at 60 Years of COBOL and the Mainframe
- 'Captain COBOL' Tom Ross on the Evolution of COBOL on Z
- Michelle Yeager and Hayley Owens on COBOL Programming
- Closing the COBOL Programming Skills Gap

This e-book was published by

TechChannel

901 N. 3rd St., Suite 195, Minneapolis, MN 55401 // (612) 339-7571

Staff list

Publisher: Mari Adamson-Bray

Senior Content Director: Evelyn Hoover

Senior Editor: Keelia Estrada Moeller

Art Director: Jill Adler

Project Manager: Noelle Heaslip

Audience Development Director: Linda Holm

Account Executive: Kathy Ingulsrud
(612) 313-1785 // kingulsrud@techchannel.com

Account Executive: Nicole Johann
(612) 336-7675 // njohann@techchannel.com

Account Executive: Darryl Rowell
(612) 313-1781 // drowell@techchannel.com

TechChannel

TechChannel.com is home to a variety of content to help you get started on your COBOL education journey.

Learn more now

SPONSORED BY