REFACTORING MAINFRAME CODE TO ACCELERATE DEVELOPMENT

Keeping mainframe development moving at the speed required by the business can be a challenge. Many monolithic applications, developed and added onto over the course of years, or even decades, are self-contained. As software development and delivery become more modernized with automation and orchestration, the cost of leaving these applications alone is growing more expensive than breaking them apart and connecting them to macro and microservices. In a time of doing more with less, many leaders are taking a second look at their root mainframe problems and seeking ways to generate value faster.

It's not COBOL, it's what has been done with it

The size of COBOL programs is often cited as a barrier to accelerating the release velocity of mainframe applications. COBOL's functionality as a programming language is why it has remained the same for half a century. It's not the language, but its implementation that is the real problem. With numerous functions housed in one program, it becomes difficult for developers to understand, edit, and compile legacy code and then test their changes. Even if they change only a few lines, they need to check out the entire program and test it, then fix anything that breaks along the way.

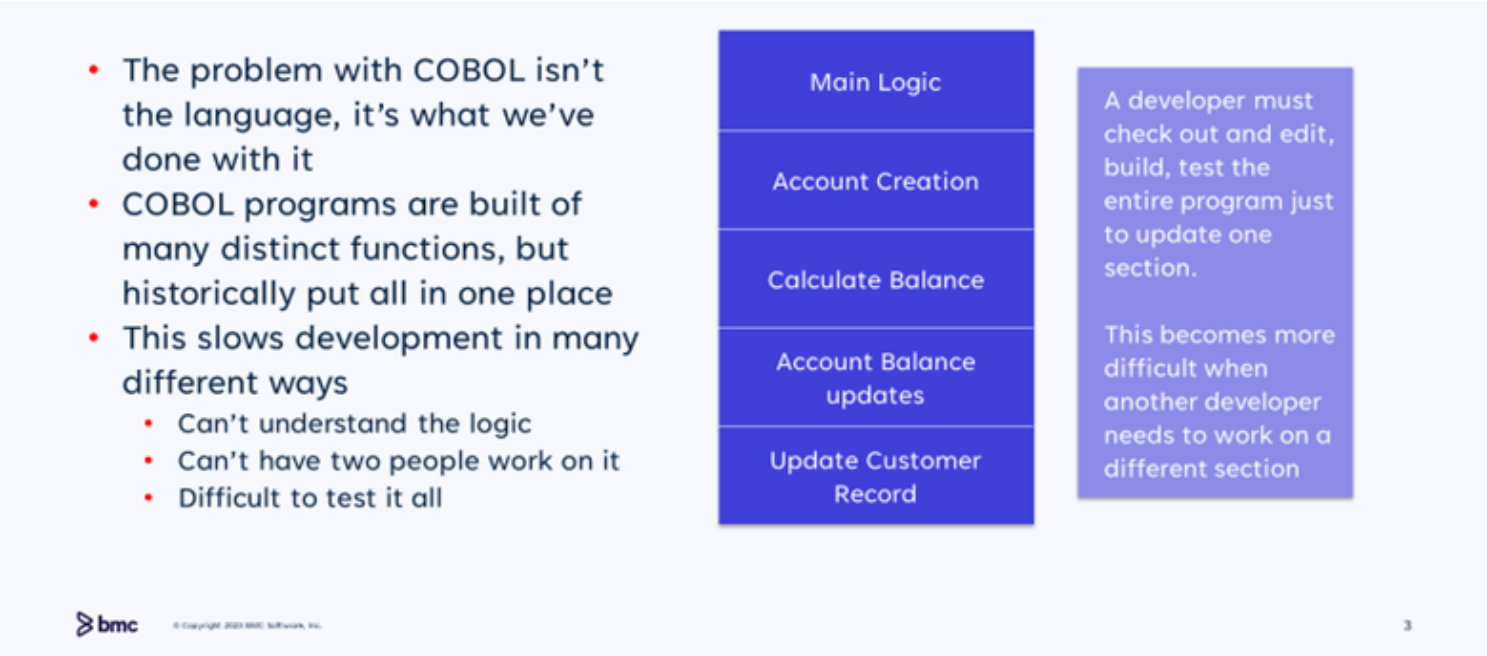

Figure 1. Why maintaining COBOL applications is a development nightmare.

Transforming the mainframe as a growth platform

Whether they're trying to unlock value of mainframe data, grow artificial intelligence for IT operations (AIOps), or enforce compliance and increase digital agility, enterprises see the mainframe as a growth platform. Most enterprises that embark on modernization journeys discover that they can't offload 20 percent of the mainframe's workloads to other platforms. But that doesn't mean they aren't optimizing the developer experience and making it easier for multiple developers to work concurrently on the mainframe. One big step forward is providing the tools to help developers break apart monoliths into separate, callable subprograms that are easier to understand, update, and test. This enables new functionality to be infused onto mainframes and prepares the way for Java- and Git-based source code management (SCM) so multiple developers can work on the same application. Being able to track a data field through the program from input to output and visualize static and dynamic interoperability, along with dependencies, logic, and syntax, helps developers understand monolithic application functions and circumvent problems, giving them more confidence in their changes and making development teams faster and more agile.

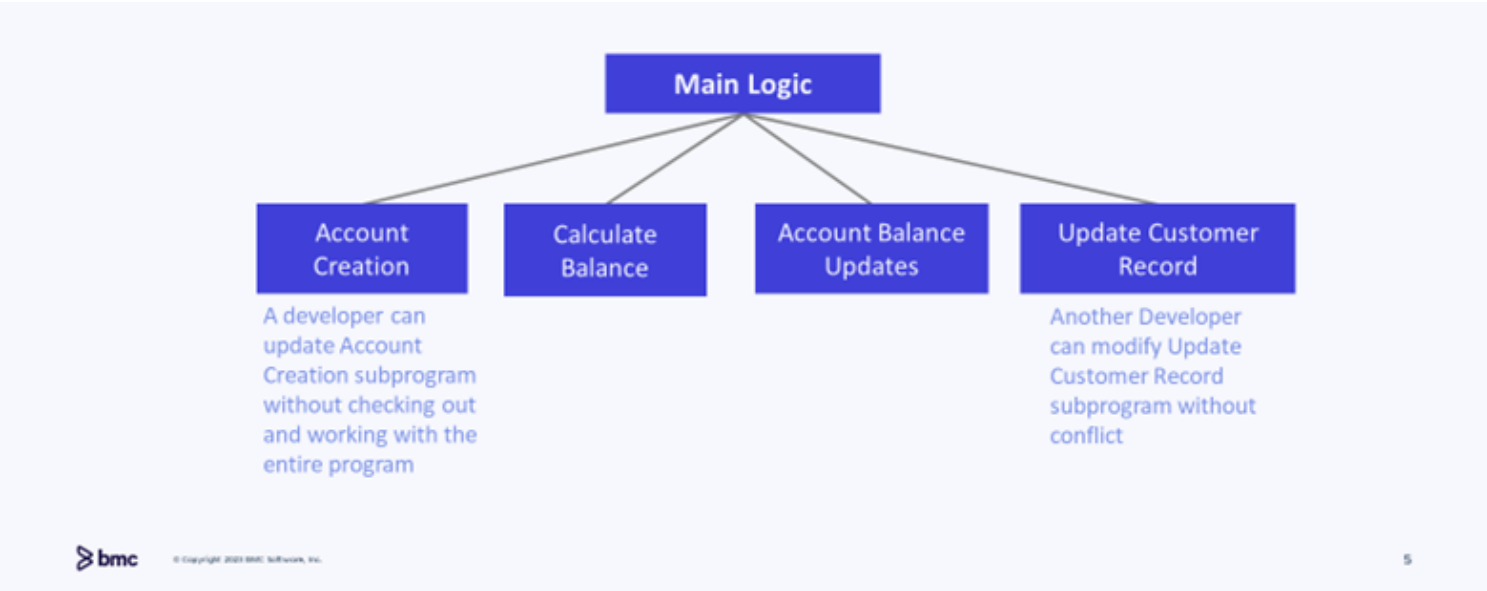

Figure 2. What if monoliths could be refactored into separate callable components? They can be.

Long-term modernization, immediate value

Modernization itself isn't fast, but there's no real reason to stay stuck with the status quo. Developers want to build on growing solutions, and the sooner enterprises transform monoliths into functional modern applications, the sooner they can curb development costs by enabling more developers to work on the mainframe. Having a diagnostic tool that can scan and visualize an application's underpinnings is critical when there are literally millions of lines of code. Seeing how an application really works in real time with runtime visualization can show developers where the problems are before they break the application.

Being able to find and comment out dead code that cannot be logically executed, no matter which data you use, eliminates a lot of technical debt as developers refactor modern applications that leverage REST APIs, Git, or Java. A lot of blame is pointed at COBOL, but even COBOL applications can still deliver value as teams undergo long-term modernizations and digital transformations.

BMC AMI DevX Code Insights (formerly BMC AMI DevX Code Analysis), available nowprovides the functionality to help developers uncover business logic and create new subprograms that are ready to compile and use. It is already accelerating the transformation of monolithic code into more modular and usable programs.

BMC AMI DevX Code Insights also gives developers the ability to identify and comment out dead code. The new Quick Fix capability, available with a right-click on the diagnostic or the identified line, will comment these lines, which prevents any future developers working on the code from wasting time by assuming those lines are valid.

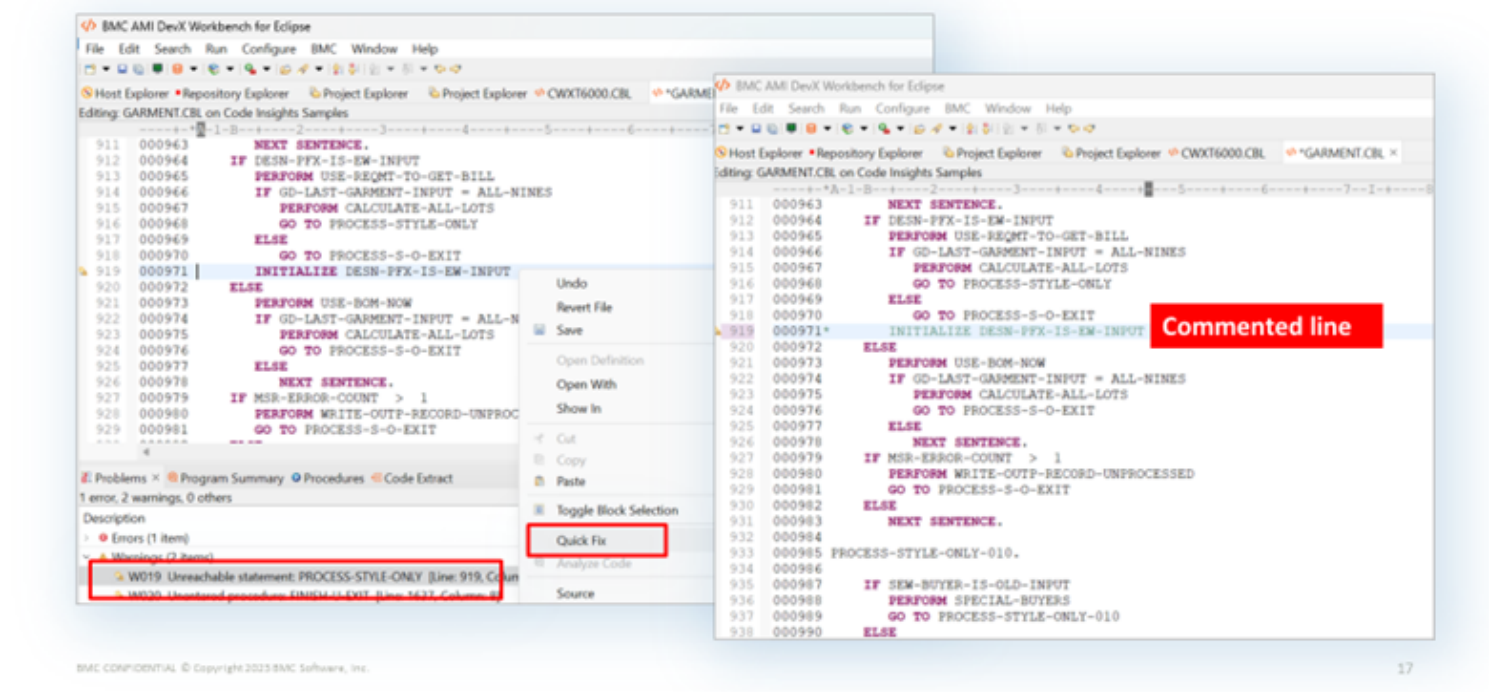

Figure 3. The new Quick Fix capability can be used to comment lines for developers.

BMC AMI DevX Code Insights is designed to help developers maintain and extend their applications with confidence by enabling them to:

- Quickly understand unfamiliar code with structure and logic charts

- Track data fields through the program from input to output
- Break apart monolithic programs using Code Extract
- Gain information on how an application really works in real time with runtime visualization

Let's not underestimate how hard COBOL development can be. It is difficult to quickly understand, make changes with confidence, and then test. Eliminating dead code—and, more importantly, breaking apart programs to manageable sizes with modern syntax—will empower mainframe application developers to deliver quality code, faster.

Listen to the Modern Mainframe podcast, "Refactoring Monolithic Mainframe Code," to hear how IT leaders are thinking about modernizing mainframe code and the tactics they're using.