

Managing COBOL Programmers in a Java World

Leveraging Wisdom of the Past to Reap the Promise of the Future

By Gary Beckinbaugh

Today, many applications in the banking industry and government still operate on large COBOL-coded mainframe systems. Some of these systems have been in place for 20 or more years, and as such, have lived long past a normal lifecycle. To extend the life of these systems, financial institutions have repeatedly augmented them with “wrap-around” functionality, most often built upon new technologies such as Java. This allows for enhanced look and feel and brings together data from multiple systems. Furthermore, many institutions are engaging in complete system rewrites using new technologies.

This shift to new technology is requiring organizations to recruit and hire a new generation of programmers versed in future-focused, object oriented architectures. At the same time, the need to support the legacy systems and retain system knowledge and domain expertise remains a critical priority.

The intersection of these two technological approaches – one traditional and well-documented; the other forward-looking and open-ended, allowing for iterative benefits but requiring a new set of coding and architecture skills – presents a challenge for organizations. How does the institution manage its legacy core systems and continue to leverage the talent of its COBOL programmers – usually longstanding employees with vast domain expertise but little-to-no experience with the new technology – in a Java world?

Different People, Different Worlds

There are certain fundamental differences between COBOL and Java programmers. Long the standard for large core systems, COBOL has been around for quite some time, leading at least one reporter covering the Pentagon’s issues with its own COBOL mainframe accounting system to refer to it as “the ancient Sumerian of computer languages.”

The vast majority of COBOL programmers has been working with the language for decades, and know it very well. Unfortunately, these experts are also a finite resource that is no longer being replenished. Many are approaching retirement age, and many more are locked into a traditional career path.

In fact, most Computer Science degree-holders who have come out of school in the last fifteen years have never even encountered COBOL programming in their studies or professional lives. This is all the more reason to embrace and retain COBOL skills and find ways to challenge these programmers in the current technical environment. To compound the issue, many COBOL programmers have little experience with newer languages, and nearly as many have little inclination to go down that path at this point in their careers.

On the other hand, Java is new, open-ended and in constant development. Programmers entering the work force are well versed in its architecture and methods for developing functionality in iterations that bring quicker term benefits than the end-to-end systems design often employed by COBOL projects. They understand critical

components of new development, such as visual modeling, agile development and the fact that in today's rapidly evolving business environment, you often have to start programming before you know all your requirements.

At the same time, though, this new generation of programmers has no experience dealing with the existing core systems so many institutions still rely upon, and with which the new technologies must interact.

An Old Standby

As institutions move into greater uses of Java (and other object-oriented languages) to expand business functionality and take advantage of new advances, some have incorrectly assumed they can simply reassign their COBOL people, transitioning them to the task of building out Java applications. However, they have quickly realized that the knowledge gap between these two populations is far greater than they may have imagined.

COBOL programmers are extremely proficient and focused on the COBOL code. Because of the nature of the COBOL programming environment, these experts tend to have a relatively limited look into the overall processing of a given application. In fact, COBOL programmers have generally not needed to interact extensively with the runtime environment or to be concerned about the deployment and how it interacts with other aspects of the software stack. That being the case, simply teaching COBOL programmers the Java programming language isn't enough.

Programming in Java demands a significant amount of associated understanding that is sequestered inside the mainframe code, as it were, and is not as relevant to COBOL programmers. Operating systems, Web servers, database servers and more must come into play in order to interact with new databases. In the COBOL-coded world, interaction with data, code promotion and interaction with the operating environment is automated by way of COBOL's very mature development state.

There are ways for these tasks to be accomplished within COBOL by design, requiring little additional input on the part of the programmer. Java requires much more current and adaptive skills to perform these same tasks.

Entirely repurposing COBOL architects into Java architects requires extensive training. It can be done, but does require patience and a good plan. A far more effective approach has been to take the valuable business and domain experience that COBOL programmers have and find ways for them to interact with and contribute to new Java development.

Finding New Roles

There are several approaches institutions can take with regard to COBOL programmers and their migration to a Java architecture environment. One is to bring in very senior-level Java architects to design and build a new framework and publicize that well within the organization. A certain segment of COBOL programmers will be willing and suited to initially becoming junior-level Java programmers willing to work within that framework on a narrowly focused area as they learn Java.

These skilled COBOL programmers, stepping into new waters, are not required nor expected to understand all of the pieces in the software stack right from the start. Again, you can't transition people straight from COBOL to Java. There should be a mentoring process, where senior level, experienced Java architects can guide willing COBOL programmers in working within that well-publicized framework so they can be productive and effectively contribute their expertise.

There are, of course, non-technical issues that are also in play. The average COBOL programmer has a significant tenure in the programming language, and it has probably been a long time since they've encountered a problem they didn't know the answer to. Finding this information in a Java world is an entirely different affair, and one which makes ongoing mentorship of Java-transitioned staff all the more important.

In contrast to the new breed of Java programmers, who are always ready to move onward and upward to the latest technological advances, some COBOL programmers are comfortable with their knowledge and may not want to transition to Java. A certain subset of COBOL programmers, then, should remain devoted to performing the important ongoing COBOL work, supporting existing systems and helping to facilitate integration with new Java applications.

Also, most long-term COBOL programmers within an organization possess a tremendous amount of domain expertise and should definitely be leveraged as subject matter experts. They clearly understand the business requirements the systems are enabling, and have years of experience supporting the current systems and client base and in dealing with unforeseen issues along those lines.

In addition to enrolling these programmers in a mentoring program to bring them up to speed on new Java development, these experienced COBOL programmers are the perfect liaison people for any integration between legacy and new components.

Never Underestimate the Value of Talent

One of the key values of moving to a Java architecture environment is the ability to produce incremental deliverables to production. It allows an organization to solve small pieces of the overall problem over time, moving into production and eliminating risks one at a time. This is in stark contrast to COBOL development, where the goal is to design and fine tune a system from end-to-end, trying to solve all issues up front before going live.

In order to achieve incremental development benefits as part of a major initiative, the organization must actively acquire the talent to do so. Growing seasoned Java talent during a major undertaking is nearly impossible. Some level of the core capability must be acquired to start with.

Likewise, the organization cannot discount the knowledge possessed by its people in current positions. Domain expertise is crucial to producing optimal results. In fact, anytime there is an interface or integration with an existing COBOL platform, a COBOL programmer should lead that initiative - even if the working components will be coded in Java - since they know that platform best.

Even though COBOL programmers may not know the language, nor completely understand how Java programmers will build a particular technical interface, they do have that crucial domain expertise. They understand the customers, their processes and goals. They understand the business problems and the issues which were encountered when the current systems were built the first time around.

As such, the need for COBOL programmers continues to be a long-term need. Corporations must keep these talented employees engaged and productive in what is becoming a Java world. A newly-hired Java programmer, regardless of the depth of their experience with Java and the latest tools and techniques, simply will not have that valuable store of domain expertise. Given the relatively rapid movement of Java programmers from project to project, or even company to company, it's not a given that they will develop that expertise either.

Financial institutions will do well to continue to tap the expertise of its COBOL talent, even while moving forward to a new architecture for future growth. In a nutshell, any organization must have the right combination of people and talent to succeed.

While bringing in people with seasoned Java skills is important to new initiatives, it is also critical to remember the inherent value of the people who have been on the ground, satisfying customers and handling business problems every day for many, many years. These two skills must be recognized, respected and deployed with equal focus to maximize the company's effectiveness.

About the Author

Gary Beckinbaugh is Senior Vice President of Product Development, Mortgage Servicing Division with Lender Processing Services (LPS)

Mr. Beckinbaugh can be reached at gary.beckinbaugh@lpsvcs.com.