CIC Commentary: Modernizing COBOL applications and application development

MOVE cobol TO future.

A Creative Intellect Consulting Commentary
This CIC Commentary has been prepared for Micro Focus as an overview of the technical dimensions of modernizing COBOL applications and the application delivery process itself. The report is aimed at development managers, senior developers and application architects who are faced with what has, for many years, seemed to be an intractable problem: how to preserve the investment in existing COBOL applications while taking advantage of the latest technologies such as mobile access and cloud deployment. We describe a migration strategy that produces a target application ready for enhancement with state-of-the-art capabilities on a mainstream platform that is strategically sound for meeting future challenges.

Paul Herzlich, Principal Analyst, Creative Intellect consulting

August 2012

Creative Intellect Consulting is an analyst research, advisory and consulting firm focused on software development, delivery and lifecycle management across the Software and IT spectrum along with their impact on, and alignment with, business. Read more about our services and reports at www.creativeintellectuk.com
Summary messages

**Doing nothing is not an option**
Pressures to evolve your COBOL applications are coming from everywhere. Your business users are clamoring for new capabilities that are only available on new technologies. Support for your hardware and middleware environments is expiring. You see that your tools for improving the development process and management are weak in comparison to the state-of-the-art. Your organization is getting less for more, when the world is going the other way. Sitting tight is no longer an option. You have to find an answer to the question, ‘what do we do about our COBOL?’

**Starting over is expensive and risky**
Starting over is an option, but one which is certain to cost a fortune and expose the business to significant risks. Rewriting and adopting application packages are clearly options for modernizing: but if developing your COBOL application was a major effort many years ago, imagine what it would entail to reproduce the finely tuned business rules embedded in your application today? Even approaches that purport to ‘re-learn’ an application from code would require serious intervention from your business users and a major testing effort. These approaches will involve unacceptable levels of risk and disruption. You’ll prefer an incremental approach.

**Front ending your applications is only short term**
One typical incremental approach has been to build a new front end to a legacy application. You treat the back end as a black box and try to introduce new front-end features with as little impact on the black box as possible. Although this works for a while, it has overheads and constraints, and does not provide a sound basis for the future.

**Migrating to a new platform is possible**
Porting your COBOL to a modern platform is possible with Micro Focus Visual COBOL. Your existing code can move with little or no code change to .NET or JVM, managed or native code. Once the existing capabilities are deployed on the new platform, they become a launch pad for future work – re-engineering to make use of the latest technologies, integration with components and services in other languages, enhancement with new capabilities, and redeployment in the cloud.

**COBOL has major advantages and a bright future**
Sticking with COBOL is not just about a safe migration onto a new platform: you also preserve your investment in your staff’s COBOL programming skills and business knowledge. Meanwhile, you share the same Eclipse or Visual Studio platforms with other programmers, effectively helping new programmers pick up COBOL easily. In the future, you continue to reap the benefits of COBOL as an especially readable language that is exceptionally well adapted to commercial systems.
The COBOL conundrum

Grappling with a seemingly intractable problem

Our lives as developers and managers of computer software systems are shaped by the businesses we serve and the technologies at our disposal. Nowadays both change constantly and rapidly. You – the development manager, the software architect or the senior developer – are your organization’s leaders in setting a course through the changing technological landscape. In rare moments when you’re not firefighting, you keep an eye on the latest trends and muse how you might take advantage of new solutions.

However, if you work in a business more than 20 years old, there is a strong chance that your musings on your path towards modernizing your IT come to a grinding halt at the same place each time. It is the moment you ask yourself, ‘what do we do about COBOL?’

Here’s the position you find yourself in: You have a hub of COBOL systems that represents a huge historic investment. Often they are well adapted to your central business processes – certainly parts of them represent detailed knowledge of business rules that even your business users may not fully comprehend. However, you know that the time really has arrived to get to grips with what seems like an intractable problem.

A widening technology gap stymies innovation

Your long-serving application has done – and is still doing – a great job for the business, but the pressure ‘to do more for less’ inevitably comes from somewhere. You know your business partners are updating their interfaces. Your business users’ wish lists get longer by the month. The competition has applications that bring their businesses closer to their customers, reduce operating and production costs and improve their competitiveness.

Figure 1 – Burgeoning business pressures and technology opportunities

The technology gap seems to be getting unsustainably wide for you to match them. New capabilities depend on a raft of open technologies for interoperability, data sharing and communications that are worlds away from
the proprietary and fairly unique nature of COBOL. Your really old systems rely on non-relational databases and ‘TP monitors’. Your data types look nothing like those in C, Java or any of their modern relatives.

Put aside ambitions to be state-of-the-art; your applications are probably lacking or are even incompatible with only moderately recent technologies that would allow you to participate in the interconnected world of composite applications, like object orientation, SOA, web services and XML.

And it is not just a matter of application capabilities. It also affects deployment and service management. Your technological isolation makes it impossible to take advantage of the enormous benefits of moving to The Cloud.

**The costs you face even if you had to change nothing**

Forget about whether you can make use of the latest shiny technological tricks. You realize that living outside the technological mainstream is costing you in a whole host of ways:

- You’re at risk while you depend on obsolete hardware and unsupported system software, databases, and middleware.
- You maintain separate development tools for COBOL software development with a raft of side effects: separate licenses and less capable, less integrated and less productive tools than those available for mainstream languages.
- You – and perhaps even your IT service partners – find it difficult to recruit staff; COBOL is becoming a niche skill and lacks pulling power for new recruits.

**What’s the solution?**

Adopting a new architecture, platform, technology or process has a cost, entails risk and takes time. But there are several ways of going about these and some solutions must be better than others.

For many managers, the straightforward solution is to rewrite COBOL applications in a trendy language on a modern platform, or to replace them with a package such as those from Oracle or SAP. Rewriting or replacing are both valid, but can be expensive and highly risky options.

The conundrum you face is this: at the heart of your business, your existing COBOL applications are central to its continuity, but they have become a drag on your flexibility, responsiveness and market competitiveness. How can you preserve the value of your core systems, but modernize them with the least risk and cost?

**What’s new about modernization?**

**Application modernization**

The term ‘modernization’ is one of those buzzwords that gain a lot of currency at a particular moment. Clearly the idea is not new. We’ve always had to maintain and upgrade our systems to keep them ‘up-to-date’. When the business changes, we do maintenance. When we run out of processing capacity, we expand – not only the hardware, but also the operating systems and middleware. When the data center evolves we upgrade with it.

Beyond this steady stream of enhancements, what more do we mean by the term modernization? A few things stand out:

- The internet for global access was a revolution that has made it possible to reach customers, employees and business partners directly and at low cost. The resulting boom in self-service applications, access anywhere and shared services has eliminated huge business processing costs.
• New technologies – mobile devices and Big Data are examples – make it possible to offer new products and services and to manage your business more effectively.

• New paradigms for client and employee engagement – such as social media, collaboration and app store distribution – are expected for improved communication, ease of contribution and democratized participation.

• Cloud deployment can transform operating costs and bring new levels of speed, flexibility and responsiveness to IT service provision.

IT has become a crucial tool in lowering costs and improving your power to meet your customers’ and employees’ needs. Automated support for your core back and front office business processes alone is no longer sufficient. We have reached the point that software not only runs the business, but often it is the business. It is what you sell and what sells you to the market. Leading edge software is a ‘must have’ in order to meet and outstrip the competition. It represents a new level of criticality for your software applications.

Modernization is the drive to get your applications on to the up-to-date technology platforms that have become essential to competing today, and growing and evolving in the future.

Modernization strategies

Some companies can justify the risk and expense of rewriting or replacing their legacy applications wholesale. An organization might choose such a strategy in conjunction with a complete overhaul of their business processes. However, in order to avoid disruption, they most choose a more incremental strategy.

The black box strategy

A typical stepping-stone to modernization involves wrapping existing core business systems in modern interfaces and treating the legacy systems as if they were a black box. You write an interface that transforms data and commands between new and old technologies.

This partitioning of client-facing front end from back end support for internal business processes is not a new architecture, but it happens to correspond to a current view, propounded by IT guru Geoffrey Moore, of where organizations are making IT investments. The new technologies are transforming the front ends, which he calls ‘systems of engagement’. This is where companies need to invest. Meanwhile, the back end ‘systems of record’ such as accounting, payroll, logistics and so forth need very little further work.

Figure 2 – Systems of record and systems of engagement

The black box modernization strategy appears to fit this model of IT investment perfectly. Companies use the approach widely but it has architectural limitations.
The distribution of logic across front and back ends is not optimal; often business logic has to be duplicated. Once it is duplicated, you have duplicate maintenance and a new risk of errors when new and old logic mistakenly diverge.

The timing of back-end processes – many of which are batch – doesn’t match the interactive nature of the modernized front-end. The black box implementation may appear awkward or incomplete.

Without extensive back-end changes, you are still largely limited to data catered for in your back-end data model.

It can also place extra burdens (which means costs) on IT processes. You multiply the development and test environments, skillsets and tools required. And although in theory you keep your core applications as they are, often they still require modification to support new systems of engagement. For example, if you add mobile access, you may need to extend your legacy system to cater for new locational data. All this adds complexity, effort and risk to coding, integration and testing.

Despite the potential shortcomings, this form of providing ‘modern’ computing features, without actually modernizing applications, has allowed many companies to extend the life of legacy systems. However, as the gap grows between new and old technologies – and the costs grow of maintaining legacy technology in parallel with newer platforms for wrapper applications – this strategy loses its potential to provide any competitive boost.

The migration strategy

In our research we find that organizations that have waited until now to modernize are risk averse and are highly satisfied with their existing applications. What makes sense to them is a two-stage approach:

- Port the existing application to a new platform.
- Incrementally re-engineer, restructure, redeploy and extend applications to take advantage of the new platform capabilities.

A new platform in this context is Internet (both in and out of browser), Windows and UNIX, along with the de facto standard environments and frameworks: Visual Studio, .NET, Eclipse and JVM.

Strictly speaking, it is not necessary to port the application to the same language. A language is a language – just a way of expressing a program’s functionality as a set of instructions. The essence of the porting activity is to achieve something of (nearly) the same structure that produces functionally equivalent results when it executes.

However, migration is more than transferring source code from one platform to another. Applications also have files, databases, transaction monitors, messaging middleware and screen presentation components that need to find equivalents on the target platform.

By virtue of changing platforms, the porting stage almost always includes a basic level of modernization including getting the application running on up-to-date screens and middleware. Once this base is established, you can continue in several directions:

- **Re-engineer** an application’s components using current technologies like HTTP, HTML, CSS, SOA, web services, RESTful web services; change from indexed files and legacy network databases to modern relational databases; deploy as .NET or JVM managed code.

- **Restructure** using static analysis and ‘slicing and dicing’ techniques to identify and extract business transactions and objects and create object-oriented code built around business objects; share objects and services.
• **Extend functionality** with completely new capabilities such as mobile access.

• **Re-deploy** in virtualized environments and The Cloud.

There is no set order to the stages after porting your application to a modern platform. Your next steps will depend on your business priorities. The key to this strategy is the existence of a target platform that is easy first to port to, and then supports all the stages that follow. It is an advantage that each stage delivers demonstrable business value.

**Application delivery modernization**

Discussions of modernization tend to focus on the technology of application software itself. But the tools and process by which application software is delivered – the application development lifecycle – can be equally instrumental in producing the same benefits: cost reductions and improved IT responsiveness.

Modernization of application software has an important side effect: you also move coding and testing onto a mainstream development environment, where you can use state-of-the-art development, testing and development management tools. Key advantages of a state-of-the-art development platform include:

• Co-existence and interoperability with applications in other languages; you can take advantage of common frameworks and managed code, object libraries, shared services and automatic security features such as code signing.

• Availability of the best and latest tools for source control management, continuous integration, automated build, and automated testing at unit, system and service levels.

• Integrated static and dynamic analysis tools facilitate refactoring and restructuring.

• Team development environments are well suited to agile development process.

Beyond the advantages specific to coding and testing efforts, modern development tools and team development environments also provide the foundation for modernization of IT management. Not only do they support the latest point tool capabilities, but they also collect information about development and testing that can be stored in repositories and reported on in management portals.

A mature, integrated ALM environment has a variety of applications difficult to achieve other ways:

• Gives development, service and IT management objective information about progress and quality.

• Provides a rich environment for teams to communicate while working together.

• Allows teams to discover and understand assets for reuse.

• Develops a profile of code quality and technical debt over time.

• Breaks down silos between development and operations (DevOps).

State-of-the-art integrated toolsets have all the information gathering capabilities and links you need to feed into an Application Lifecycle Management (ALM) suite. Such suites can transform the economics and effectiveness of overall IT management. However, they rely on your applications being on a modernized environment. Essentially, development tools for modern environments make your development process much more ALM-able than it is in an un-modernized environment.
COBOL for the future

**Porting COBOL to modernized COBOL**

The migration strategy for modernization outlined earlier is agnostic about target language. However, if it were feasible to port a COBOL application directly to COBOL on a modern platform, you would improve the strategy by reducing the risk of changing languages. Even with that seemingly obvious benefit you might still want to consider whether it is the right language for the future.

**Why stick with COBOL?**

Although you know that any migration is more than just a question of just porting programs, code is still the most crucial element. It is the embodiment of business rules built up over many years. It encapsulates knowledge of the business – algorithms, calculations, restrictions, and data domains – that users may have long ago forgotten and which would take extensive research to re-discover.

Another factor is that, putting aside a few dusty and hidden corners in the application, you and your teams know and understand it. You have experience of how to tease out the code’s secrets; you recognize the data name conventions; and simple commands like MOVE and PERFORM mean so much more to your programmers than languages where ‘=’ instead of ‘==’ or ‘===' can mean disaster.

However, the principal reason for sticking with COBOL is that it’s what you already have. Keeping the COBOL code is just one less thing to worry about in a migration:

- To the extent you can carry across code and files without modification, you reduce the likelihood of introducing migration errors and can tailor your test strategy accordingly.
• You preserve the inherent advantages of COBOL, such as its readability and suitability for financial calculations.

• Best of all, you preserve your experience, your skills and an application that supports the vast majority of what the business needs.

**But is it right for the future?**

If you were choosing to write a mobile application today would you choose to write it in COBOL? Probably not. In most cases, it’s not the language best suited to the problem. However, COBOL is just the right language for the sort of information processing problems found in commercial applications, with their record structured data, transactional orientation, and high transaction loads. Those are the problems you are still trying to solve in your modernized application.

When it lost its street credibility in the 1980s, the main criticism leveled against COBOL was verbosity, and it also had some glaring restrictions and omissions that make no sense in the world of massive desktop computing power.

But COBOL has continued to evolve. While it remains a command-orientated language — which still helps readability and maintainability — fixed A/B margin formatting, uppercase-only text and rigid division-section structure are things of the past. It has acquired modern-language features such as object orientation, user functions, intrinsic functions, recursion, dynamic tables, function pointers and long literals.

**Figure 4 COBOL as a balanced modern language**

COBOL can seem otherworldly to those used to the obscurity of bracket-and-brace languages like C, Java and their derivatives, but it represents a reasonable trade-off between fitness of purpose in commercial applications, readability and compactness. In other words, COBOL’s past shortcomings don’t compromise its appropriateness for the future.

**COBOL back in the mainstream**

Given the other benefits, the question of preserving your COBOL code as the target in a migration for modernization finally reduces to whether COBOL can be a full participant in today’s two mainstream development environments: Visual Studio and Eclipse.

The option for building and running COBOL applications on PCs and UNIX machines has been available from the early PC days. IBM did PC-COBOL; Microsoft did MS-COBOL. RM COBOL and Micro Focus COBOL were pioneers too. COBOL never took hold as the right language for native PC-based applications, but PC- and UNIX-based COBOL did find a market for offloading mainframe development. COBOL also represents a
significant portion of the code in distributed systems that have been built over the years on client-server architectures.

Micro Focus’s latest Visual COBOL product takes this lineage a step further. You develop, compile and test COBOL in Eclipse and Visual Studio. Your application can be either native or managed code. The compiler accepts a large number of COBOL dialects and implements a vast range of COBOL data types, so there is very little adjustment to code required in a ported application.

Most significantly for the future, COBOL no longer needs to be treated as a legacy maintenance language. Visual COBOL provides integration with nearly all the features of the mainstream development environments. With that you get interoperability with code, objects and services in other languages. This means that once your COBOL application is migrated onto Visual COBOL, which can be done with a minimum of change, you can take your time to add the up-to-date capabilities you’ve been thwarted in building until now. You’ll be able to do that in a COBOL with a clear future and diversify into other languages where technology or business processes requires it.

Finally, it also helps you deal with one of the most difficult to manage resources - your people. With COBOL as a .NET or JVM language in a familiar environment, you bring COBOL development closer to a new community of programmers. Your existing COBOL programmers enjoy a major skills upgrade, while you preserve the investment in their COBOL programming skills and considerable knowledge of the application and the business.
Conclusion

Years ago, COBOL programming was the mainstream of application development, but the PC and then the Internet changed that. Although COBOL development never declined, the development mainstream moved elsewhere. Pressures have now built to the point where continuing to support COBOL applications in legacy environments is a drag on organizations’ abilities to compete.

Micro Focus Visual COBOL can be the cornerstone of a modernization strategy that preserves the business value of existing COBOL applications and the inherent advantages of COBOL, while also positioning future development and maintenance on today’s mainstream development delivery platforms; Visual Studio and Eclipse.

New developments in COBOL programming environments mean that it is time to re-assess your options for modernization. If your thoughts about the future of your applications are snagged on the question, ‘what do we do about our COBOL?’ then:

- Understand that it’s time to move beyond the stage of doing nothing. The business climate is forcing your hand.
- Evaluate and compare the costs and risks of migrating your COBOL application to the cost and risks of other re-engineering approaches or replacement approaches.
- Consider the fact that COBOL can be more than a maintenance language. It can be the sound foundation for the future of your applications, your staff and the management of IT.
- Plan how you and your colleagues can make the most of your existing COBOL skills, while extending your expertise into the latest, state-of-the-art technologies.

The tools are now at hand to modernize your COBOL applications, IT management processes and professional skills in an efficient, low-risk way that lands you on sound platform for the future.