COBOL AT 60

The Legend Continues

Brought to you by MICRO FOCUS

Introduction

The perpetual drive for technological evolution can produce less-than-essential results, gimmickry rather than genuine innovation. We can all think of previously must-have technology that has been Darwin-ed out of our lives. In contrast, bona fide game-changers not only endure, but continue to evolve in of themselves. Google is 21 years old, and it has moved on from putting the “hasta la vista” into AltaVista, into a multitentacled, life-organizing digital being. Now 25, Amazon left peddling cheap paperbacks online behind to become the world’s largest e-commerce and cloud computing platform.

But despite these timelines, and the brutal, ephemeral world of technology, their constant reinvention means that no one thinks of Google or Amazon as old. So what does that make COBOL and IBM CICS, 60 and 50 respectively? For COBOL, it’s tempting to point to NASA launch codes, again, to prove ongoing roadworthiness. But as we’ve discovered, for technology, the past is another country and a sentiment-free zone. To endure, tech must remain relevant. So, rather than reflect on a solid, largely unrivalled track record, I want to look forward and explore what the future looks like for COBOL.

Digital Disruption

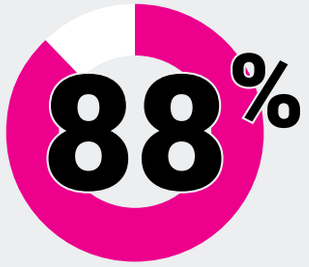
The technical innovators' work surrounds us. They are building API and microservice capabilities, developing new language choices like Python, R, Ruby and JavaScript. Entrepreneurs are trialing blockchain, digital currencies and DLT for finances. New use cases are being tested using artificial intelligence (AI) and robots to automate everything. But new doesn't mean progress for all. Organizations are facing unprecedented competitive pressure, and the evidence is there. Only 12% of the Fortune 500 from this decade had survived on that list since the 1950s. Adapt or perish. Back to Darwin. It really is as simple as that.

Like ripples in a pond, disruptive change radiates out, affecting everything. IT systems are not immune, but some, regardless of circumstance, must not fail. They are too important, strategically or financially, to be swapped out, cut loose and replaced by something else. The portents are not good. Based on a 2014 survey of 3,300 IT decision makers from mid-size to enterprise-size businesses in 24 countries for EMC, Vanson Bourne found that data loss and downtime cost enterprises more than $1.7 trillion in the last 12 months.[1]

1. Vanson Bourne, jabtechus.com/hello-world/

Such business-critical systems are often written using business-centric, more established technologies. The platforms might be mainframes or other robust servers, whether Linux, UNIX, Windows or the cloud. The data layer either reliable, rapid flat-file structures like VSAM or industrial strength databases like Db2. Transaction monitors? CICS, IMS. The application language? Never really any debate. COBOL: reliable, ubiquitous, trusted, oft-misunderstood, yet fantastically successful. And in 2019, with 85% of all applications regarded as "strategic," COBOL quietly goes about its business of adding to 60 years of active service within the IT industry.

So while the waves of change threaten to swamp some, core business systems written in COBOL continue to provide the functionality many organizations rely on. But what about what's next? Ticking over isn't moving forward. And as we've established, risk-averse enterprises view their applications as too important to waste and too valuable to fail. The good news is that the bedrock can also be a springboard. These systems are extremely well-placed to launch low-risk innovation. So pragmatic IT leaders are delivering new user experiences, business facilities and



FIRMS IN THE 1955 FORTUNE 500 THAT ARE GONE (AS OF 2014) —American Enterprise Institute

$1.7 TRILLION

COST OF DATA LOSS AND DOWNTIME TO ENTERPRISES IN THE LAST 12 MONTHS —Vanson Bourne

competitive differentiation by innovating, enhancing and integrating core systems with contemporary technology.

With so much change needed so quickly, there is no time nor need to throw away what's already providing value.

Using technology that works–COBOL applications–as the innovation baseline guarantees the future relevance of this time-proof language for years to come. And that's a view shared by those who continue to use it.

The Only Constant Is Change

The ripples of technology change offer near-limitless opportunity for IT innovators able to ride the wave of powerful tooling, platforms and technology. Such as? The drive toward an API economy, using microservices, deploying into containers, running on serverless environments, collaborating in open frameworks using DevOps to deploy new facilities that offer enhanced user experiences and solve real business challenges, using ostensibly new technology.

Let's consider some scenarios faced in IT teams. It's a wish list of "we want to":

- ✔ Unlock the business functionality embedded with a COBOL system to reuse it as a component, service, object, microservice or whatever
- ✔ Change our underlying data model but not the core applications
- ✔ Provide a more user-friendly customer experience to an existing application
- ✔ Integrate new third-party applications with our core business systems
- ✔ Deploy some business applications in the cloud

These disparate requirements represent innovation in the context of current core business systems, building in new capability from a position of established strength, of power, rather than razed earth. In each case, the COBOL application is being redeployed or reused, according to the technical use case. The tooling is there, it merely requires the awareness and the right approach. You might call it a modernization strategy (I certainly would).

COBOL's Credentials

COBOL once had few competitors, and "portability" just meant supporting the only computing option at the time: a handful of what became known as mainframe computers. This modest beginning sparked a revolution that has launched a thousand technological advances and IT choices for users. Examples? Consider the following random list.

- Data stores: IMS-DB, VSAM, QSAM, Db2, ISAM, IDMS, Adabas, Datacomm
- Monitors/messaging: CICS, IMS TM, MQ, WebSphere
- Languages: C, C#, Java, Visual Basic
- Chipsets: Intel, mainframe, IBM Power Systems
- OSes: z/OS, VSE, AIX, OS/2, MS-DOS, Windows, Linux, Linux on IBM Z
- Managed code environments: .NET, JVM
- Cloud environments: Azure, AWS, IBM Z
- Containers and virtual environments: VMWare, Docker, Kubernetes
- Contemporary application language constructs: microservices, API, object orientation, SOA, web services

Back to timelines. Plotting COBOL's history of integration is a roll-call of cutting-edge technology, where digital innovation has added to established COBOL investments. But it's a different world now, where new technologies must integrate with core COBOL back-end systems. It's time to collect a dividend. In many cases, it means integrating mainframe-based core business systems into an increasingly-connected world. There will be questions around future innovation, integration and continued investment. But COBOL has had to deal with this scrutiny for more than half a century now. It always has the answers.

Plotting COBOL's history of integration is a roll-call of cutting-edge technology, where digital innovation has added to established COBOL investments.

Ready for the Next 60 Years?

We mentioned Google, and a quick search on COBOL namechecks comprehensive descriptions from Wikipedia, Bill Klein, and profiles of pioneers including Jean Sammet and Grace Hopper. But as we said, its rich legacy can be viewed by some in the IT industry in 2019 as a handicap. Old isn't good. So COBOL's readiness to embrace future decades is more relevant than the past.

The COBOL label, first documented in September 1959, describes a language specification that aimed to enable the noncomputer literate to communicate more effectively with computers, to meet the growing need for an "open-ended, problem-oriented and machine-independent computing language … capable of accepting change … that [uses] simple English" computing services in government and industry. Under the stewardship of Hopper, the first incarnation, COBOL-60, took shape within a year.

So why has COBOL endured so well?

For any technology to thrive and survive, history suggests it needs five core attributes. Few would argue–and 60 years of evidence confirms–that COBOL ticks all the boxes.

1. Innovation
2. Longevity
3. Portability
4. Built for business
5. Readability

Innovation

Computer languages must speak the vernacular of the changing IT landscape. So the COBOL vendor community invests tens of millions of dollars every year to ensure it remains contemporary. These investments support contemporary technology and integration with other language applications such as Java, C++ & C#.

As well as being fluent in mainframe, Linux, UNIX and Windows, past COBOL applications can pick up the lingua franca by simply being recompiled to run in the cloud, .NET and JVM. A few simple steps can deploy a COBOL application into a Docker container.

In addition, the growing use of SOA and web services, and the range of technology elements, such as XML, WSDL, SOAP, JSON/REST, HTML, ensure application integration and connectivity across the enterprise. And this is integration with a big "I" because business innovation depends on the ability for new, digital systems to hook into the critical processing facilities and data within

long-established COBOL apps. Proof (if more proof was needed) of COBOL's ability to evolve and support the growing codependence of contemporary digital technology to older applications.

COBOL continues to evolve to take advantage of emerging technology; the COBOL language standard was updated in 1974, 1985, 2002 and 2013. As future standards emerge, the language will evolve.

The COBOL software vendor community, including IBM and Micro Focus, played its part both by participating in the standards body, and also being one of the key pioneers of the new standard from a compliance perspective. It also re-affirmed its commitment to its clients, and the language, by establishing the backwards compatibility policy: Any COBOL program, anywhere, which conforms to the standard(s), will compile with the latest COBOL product.

Longevity

New application development is rarely from scratch. More frequently, client-focused innovation is about delivering business applications through new channels. The business logic in COBOL systems has been hard-won and expensively assembled. Business sense dictates it must also be extensively mined.

- **Reusability:** COBOL's highly readily understood nature is why it permeates the enterprise. Why constantly re-invent the wheel when you're already moving forward?
- **Accessibility:** Alternative development languages can rapidly access COBOL value using native semantics and data types
- **Compatibility:** The backwards compatibility policy helps to keep applications current and provide a low-risk environment for systems development, a stark contrast to other technology options where the code has to be rewritten after every compiler release change.

Such important factors spawned a meteoric growth in the usage of the language for commerce. Some estimates put the amount of COBOL code in production in the hundreds of billions. This volume of code and level of investment–especially given the importance of some of the business functions it provides–creates a lasting heritage and longevity.

Some estimates put the amount of COBOL code in production in the hundreds of billions.

Portability

Choice defines the IT industry. There are multiple answers to any question, different resolutions to every challenge. This is especially true for platforms. Most large organizations are characterized by disparate, heterogeneous hybrid IT environments: mainframes, server farms, mobile devices and every point in between.

So software vendors must make their applications available on more platforms. Clients choose a vendor's application primarily for its functionality, but will also consider many other factors, from the practical to the corporate and strategic. They include the breadth of platforms supported, relative ownership costs, user requirements, skills profiles and supply-chain policy. There's a lot to consider, from planning to consumption.

Developers working with contemporary COBOL technology can analyze, develop, debug, test and deploy their applications across every platform their clients use. Integrated development environments provide instant edit/debug cycles and feature-rich tooling. Crucially, all this good stuff is running the same portable code as new industry-leading frameworks such as .NET and Eclipse, containers and cloud. Of course, as a business-focused language, COBOL can deploy across the leading enterprise platforms. Java, supposedly the language of portability, falls short of COBOL's breadth.

Programming languages aren't valuable in of themselves. They depend on the ability of the coders to align application with business need. To contribute to the business, the language must combine a state-of-the-art environment for building robust apps, including contemporary features (Intellisense, rapid code/debug, UI builder) and be built on the latest frameworks (Visual Studio and Eclipse).

As a business-focused language, COBOL can deploy across the leading enterprise platforms. Java, supposedly the language of portability, falls short of COBOL's breadth.

Built for Business

Successful core enterprise systems share certain characteristics. They have strong, reliable IT infrastructures that offer validity, strong data manipulation, accuracy, speed and accessibility. In short, something fit for business needs, and designed with scale in mind. Consider the following attributes.

- COBOL's type-rich language enables data to be described accurately, with explicit scope and limits to meet corporate coding standards and industry-specific compliance requirements
- COBOL's ubiquity ensures consistency and accuracy across the organization and third parties, including partners
- It delivers arithmetic accuracy up to 38 digits–more than any other language. It's no coincidence many of the world's financial powerhouses rely on COBOL.
- Speed of application execution is a key criterion for core business applications. They need horsepower to deliver both computational speed and batch support. COBOL applications can be optimized, thanks to innovations from IBM and Micro Focus, for specific hardware and platforms, increasing performance and throughput.
- COBOL is renowned for its data handling, providing capabilities to deliver stronger data manipulation:
  - Faster data access than any RDBMS, and support of data files of a variety of formats (RDBMS, Indexed, Sequential, Relative)
  - Data manipulation and reporting built in to the language with the SORT capability. Uniquely, users can SORT and filter within COBOL without additional tools or steps—much faster than having to handle this outside the language.

COBOL'S BUSINESS ATTRIBUTES

- TYPE-RICH LANGUAGE
- UBIQUITY
- ARITHMETIC ACCURACY
- OPTIMIZABLE FOR SPEED
- POWERFUL DATA HANDLING
- QUICKLY LEARNED

Readability

COBOL is easily understood and quickly learned, unlike many programming languages where code is hard to understand, even for those with the skills to write it. COBOL is structured in terms of its layout, and uses active English-derived constructs ("add" is ADD, "equals" is EQUALS) that tell the reader at a glance what the code is trying to achieve. This offers tremendous benefits:

1. As anyone can write it, it becomes possible to create low-cost high-availability resource pools to construct applications and because anyone can read it, there are significant downstream benefits, too. IT can tap into a theoretically unlimited resource pool to work on COBOL systems, and there's no barrier to entry for future COBOL programming skills–a major bonus, in terms of strategic planning and investment.
2. If anyone can read it, anyone can maintain it. A second generation of programmers can code in COBOL applications they haven't originally written. Java and C# teams can review the COBOL back end to their new front-end code, using the same integrated development environment (IDE). Nondevelopers can follow the flow where necessary, QA staff can assist with code walkthroughs and debugging work, and so on.
3. COBOL's high legibility avoids the major common pitfall where coders simply rewrite something they don't understand in a language they do. COBOL typically passes the comprehension test. As explained a decade ago by a Forrester analyst[2], "COBOL is one of the few languages written in the last 50 years that's readable and understandable," a view shared by others. "It's not just a write-only language," says Michael Coughlan of University of Limerick. "You can come back years later and understand the code."

2. Mike Gilpin, analyst at Forrester research, https://www.silicon.co.uk/e-regulation/cobol-heavily-used-50-years-on-1873.

Conclusion

If a year is a long time in technology, then six decades of service in the IT world is a phenomenal achievement. COBOL pre-dates CICS, UNIX, Linux, Windows, Java, the internet, the personal computer and even the IBM mainframe.

In every SHARE event[3], the "IDE shootout" plays out in front of packed rooms. Three or four major mainframe COBOL product vendors showcase the latest innovations and tools to defeat the collective enemy, the "old way of building mainframe applications." The exercise symbolizes ongoing investment and commitment from the vendors and wider mainframe community in COBOL.

COBOL has endured and evolved because it was designed for business then, and adapted to meet its needs now. As phones and cars metaphorically demonstrate, great ideas can evolve and adapt. Smart technology does the same.

Of course, for those who believe in such things, this was all foretold. Indeed, the COBOL language has given us a clear clue all along. As valid syntax might read—

```
// EVALUATE COBOL-VALUE
// WHEN 60 CONTINUE
```

3. See share.org

START YOUR JOURNEY

Contact Micro Focus to build your future application strategy today

REQUEST A VALUE PROFILE MEETING

MICRO FOCUS®

microfocus.com

Copyright 2019 Micro Focus. All Rights Reserved.