White Paper

COBOL at 60—a Living Legend

Celebrating COBOL’s Enduring Value

Thomas Carlyle’s saying “No pressure, no diamonds” can apply to many things, but to the COBOL programming language? In the digital era where the only constant is change, COBOL’s Diamond Anniversary is the perfect milestone to assess the pressures it has endured to remain at the top of its game six decades on.

opentext™

Table of Contents

page

"Many critical systems, dating back as far as the sixties, rely on the power and stability of COBOL. However, in today's world of computing, COBOL utilises modern development environments and happily integrates with other cutting edge technologies. Truly a language for the ages."

MIKE MADDEN
Owner
Legacy IT Consultants Ltd.

Introduction

With each passing year, technology evolves and changes the world around us. 2019 is no different: this year, Google is 21, Amazon is 25, and some might argue they have changed the face of technology, and how we use it, forever. Another potent force that shaped the world of IT and commerce for decades— COBOL— enjoys its diamond anniversary this year. In an age of unprecedented pressure to innovate, where new trends seem to last only for the briefest moments, the spritely sexagenarian COBOL continues to shine.

This white paper looks at the topic through three dimensions:

- Why the world needs robust—and ever-changing—business technology
- What Innovation means in terms of technology, and talent
- The special ingredients that gave COBOL its everlasting power

Section 1. When Applications Mean Business

Why the world needs robust—and ever-changing—business technology

Digital—Disrupting Everything

This paper will assert the enduring value of COBOL as a smart bet for core business applications. We will start with a broader view of today's IT domain, and the challenges it faces.

The maxim[1] "change is the only constant in life" symbolizes the digital age. Corporate IT has to constantly reinvent itself—build new, innovate, accelerate.

The technical innovators are building API and microservice capabilities. The new language[2] choices are Python, R, Ruby[3] and JavaScript; entrepreneurs are trialing blockchain, digital currencies and DLT[4] for finances; new use-cases are being tested using AI and robots to automate everything, and driverless cars prowl the streets. Not everyone makes the cut in this brutal drive to survive. Only 12% of the Fortune 500 from this decade had survived on that list since the 1950s[5]. Adapt or perish. It really is as simple as that.

"COBOL is 60-years young. The language that powers the mainframes that run the world is as relevant today as it was in the 1960s. With the presence of new digital pressures, the mainframe and COBOL are back at the forefront for the modern developer enabling innovation and transformation."

STEVEN DICKENS
IBM LinuxONE

Refreshing Stuff

Disruptive change is great for consumers, not so much for those who must deliver it—and IT systems are at the sharp end. Innovation is risky business, with $1.7Tr having been wasted[6] on failed IT projects in a single year . But some systems simply cannot fail. Some systems are just too important to be swapped out and replaced by something else.

Such critical systems are often written using business-centric, more established technologies. The platforms might be mainframes or other robust servers, whether Linux, UNIX or Windows, and whether or not in the Cloud. The data layer either reliable, simple flat-file structures or industrial strength databases like DB2 or Oracle. Transaction monitors? CICS, Tuxedo, IMS. But, for core business apps, the language is more often than not COBOL—the reliable, ubiquitous, trusted, fantastically successful COBOL.

COBOL remains a firm feature in industry-wide application language reports (OpenText™ was included as a Sample Vendor for COBOL in Gartner IT Market Clock for Programming Languages[7]). And with 85%[8] of OpenText's customers saying their COBOL applications remain "strategic," COBOL quietly goes about its business of celebrating 60 years being the IT industry's unsung hero, providing the critical functionality relied upon by many organizations.

"For the past four decades, United Life Insurance Company, now a member of the Kuvare family of companies, has built its core life and annuity policy admin systems on COBOL… We continue to modernize and build new ways of integrating our COBOL systems with modern technologies such the cloud. Building our business and our insurance systems on COBOL has delivered real returns for the company and positioned us well for new initiatives."

JIM VEGLAHN
United Life Insurance Company,
Part of the Kuvare family
of companies

Pragmatic IT leaders are looking at ways of Innovating, enhancing and integrating those core COBOL systems to provide new user experiences, business facilities and competitive differentiation. The applications are too important to waste, too valuable to fail, and—importantly—are extremely well-placed to act as the springboard for low-risk modernization. When it comes to building and maintaining critical systems that run big business, nothing has overtaken COBOL.

Modernization Matters

The IT Modernization market is reported as growing in the range[9] of 20% per year, and that IT market analysts are seeing[10] "a shift from a rip and replace approach towards modernization strategies". OpenText's own customer survey reports that over half of the organizations asked will embark on Modernization projects in the next 24 months.

How is Modernization achieved? Importantly what is achieved is driven by objectives which is driven in turn by the necessity for change, which can in turn be based on a range of issues. So any solution must be equally robust and comprehensive. Crucially, a pragmatic, low-cost, low-risk transformation model needs to support 3 key focus points.

- **The Application**—Application modernization is fueled by customers demanding a modern user experience. That new capability also requires core processes and secure transactions to change. Using contemporary development technology, containerization, API models, managed code and service-based architectures, IT teams can create a hybrid best-of-breed model that link trusted systems and new technical innovation
- **The Process**—Service delivery is driven by what customers need, and when they want it. With the operational and technological changes associated with DevOps, the enterprise COBOL world can deliver at the speed the business needs regardless of the development or deployment platform.
- **The Infrastructure**—Because users now consume apps from anywhere, many organizations are looking for flexible deployment options to deliver their services. In a hybrid IT world of mainframes, mobile and cloud, modern platform deployments need to support secure, flexible business services for the digital age.

With so much change needed so quickly, there just isn't time to throw away what's already providing value. Pragmatic IT leaders are reusing what works and innovating from there. Thanks to ongoing technological investments by the vendor community, COBOL continues to support, integrate with and embrace technical innovation, as it has done for decades.

Section 2. Investing in Innovation

What innovation means in terms of technology, and talent

The explosion of technology choices in the digital age offers a near-limitless scope of opportunity for IT innovators. Unprecedented availability of powerful tooling, platforms and technology offer tremendous possibilities. The drive towards 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. Such possibilities exist today, and buzzwords aside, the scenarios are genuine enough and all, to a large degree, aim to solve genuine business challenges, using ostensibly new technology.

But what if the innovation needed will only really help the business if it can work with core COBOL systems? After all, that's where the backbone of today's functionality and data access resides. Let's consider some scenarios faced in IT teams, who may want to:

- Unlock the business functionality embedded with a COBOL system to reuse it as a component, service, object, microservice, or whatever.
- Ship new features more quickly than ever using agile delivery methods and technology
- Change the underlying data model but not the core applications
- Provide a revitalized, multi-device, user-friendly customer experience to an existing application
- Integrate new 3rd party applications with our core business systems
- Deploy some business applications in the cloud, possibly using containers

Such requirements are often placed upon core enterprise COBOL applications. In each case, this change, this innovation, can take place by building upon incumbent core business systems, building out from a position of strength. In each case, the COBOL application remains fundamental to the new IT capability, and can be refined, leveraged, redeployed or reused according to whichever technical use-case is required.

To that end, COBOL applications can:

- Support new technical use cases in terms of service orientation, API integration, .NET and JVM integration, container deployment. COBOL applications can benefit from creation of new user interfaces as part of modern IDE toolkits, as part of more contemporary composite application.
- Deploy across today's Hybrid IT environments across mainframe, Linux, Unix and Windows environments, either on physical or virtualized / cloud environments, or any combination
- Benefit from renewed development process covering agile, DevOps practices, integrating across a range of CI, CD and test automation tooling for faster application delivery

Contemporary, comprehensive tooling is there to achieve all these things; it merely requires the awareness and the right approach, based on the specifics of the challenge that is being faced.

> **"Modern COBOL has allowed us to re-use our existing code base and modernize it, which is really exciting for our developers. We've been able to move to the Cloud and take advantage of improved IDE as well as Agile and DevOps practices. This new approach has enabled us to bring product to market faster and respond to customers sooner. COBOL and application modernization are a fundamental part of our company's future and we're very proud to be part of that journey."**
>
> **RUSSELL HOLLICK**
> Chief Software Architect
> SYSPRO Corporate

"We have decades of COBOL experience within our company and with new tools, we can now leverage a modern IDE, where developers can work with COBOL and Java, together. Mixed language debugging helps us find problems fast, improve quality and deliver better software to our customers. COBOL is an important element of our future business strategy."

SVEN OLDENBURG
CTO
DIE Software

Innovation and Integration: COBOL's Credentials

When designed, COBOL had few language peers to concern itself with. And the concept of portability was merely the requirement to support the handful of what became known as mainframe computers that were the only computing option at the time. Of course that was the genesis of the technology revolution which saw an explosion of technological advances, and a proliferation of choices for IT users, ever since.

Consider the following:

Data Stores	Languages	Chipsets	Operating Systems	Managed Environments	Cloud Environments	Containers and Virtual Environments	Contemporary Application Language Constructs
IMS, VSAM, DB2, ISAM, IDMS, Adabas, Datacom, Oracle, SQL Server, PostgreSQL	C, C#, Java, Visual Basic	Intel, Sparc, Mainframe, PA-Risc, DEC Alpha, Power	Z/OS, AIX, Solaris, HP/UX, Unixware, SCO, OS/2, MS-DOS, Windows, Linux	.NET, JVM	Azure, AWS	VMware, Docker, Kubernetes	Microservices, API, object orientation, SOA, Web Services

This table represents an eclectic and diverse array of technologies and environments. Interestingly, if you added "works with" or "is supported by" for each, there's one answer that unifies them all: COBOL.

Plotting COBOL's integration and innovation investments reads like a history of cutting-edge technology. As the world has changed, so has the need for different types of technology to integrate with core COBOL back-end systems, and therefore such has been the investments made in COBOL technology to do just that.

Skillfully Done—Tapping into Talent

Central to any theme of innovation through reuse, is the acceptance that you have to know enough about your current state to define the right path to the future. Plus you have to have enough skilled staff available to continue to innovate using your choice of technology.

Tackling the first issue—knowing where to start—is a skills question that technology can help answer. With the best will in the world, no single developer could be expected to understand an entire application estate that might run into scores of millions of lines of code. Therefore, technology that helps understand the status quo is as vital as anything that supports the transition. Contemporary technology that enables COBOL-based core (mainframe or distributed) systems to be analyzed, visualized and understood ensures the task of "taking stock" can be achieved (regardless of the team's experience level).

OpenText's mission to offer a COBOL programming environment for all microprocessors has understood the importance of providing productive, contemporary technology for the language for a long time. Nowadays, OpenText offers products for those responsible for learning, maintaining and enhancing core COBOL applications, through its mainframe[11] and distributed[12] COBOL application analyzer products, and its modern IDE-based mainframe[13] and distributed[14] COBOL application development products.

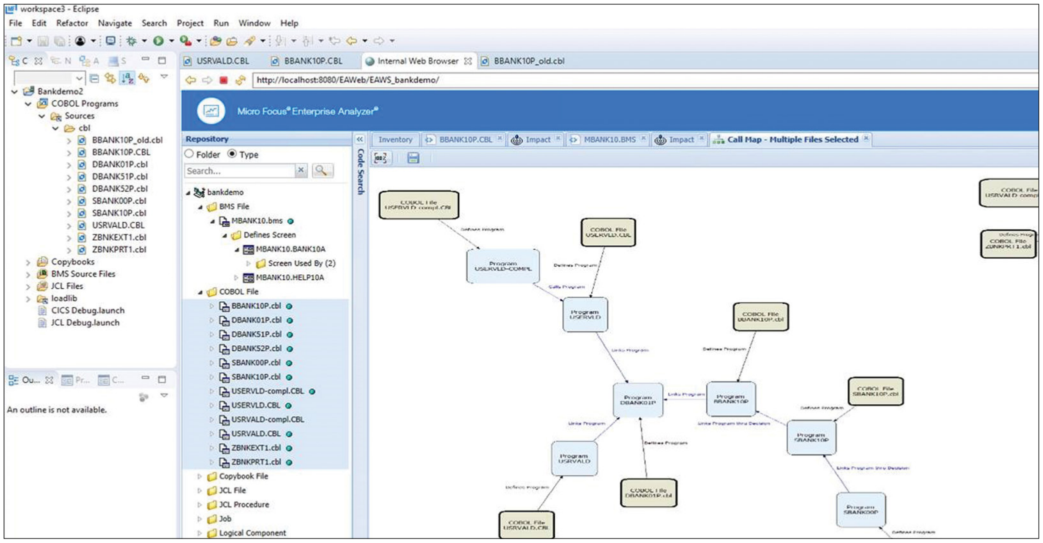


Figure 1. Contemporary Analysis Technology simplifies the task of understanding core COBOL applications

Secondly, the technology might be useful and provide the right insights for the application teams to make effective change more quickly, but you need a team in the first place. So another vital characteristic of the technology is that it lends itself to being easy to pick up by the generation of new IT graduates. These technical staff are well versed in languages such as C# and Java (and much less so COBOL), but who do have a strong understanding of contemporary IDEs, and other more modern IT principles including agile, CI/CD, object orientation, thin-client architecture etc.

While this new generation will likely baulk at traditional ISPF editors or awkward user interfaces, it will have zero concerns with Eclipse or VS based editors and tooling. These contemporary IDEs can be used for COBOL development, whether for mainframe or distributed platform deployment, just as they can for other languages, as result of continued investment in COBOL tooling.

Longer term, the task of securing the future generation of skilled, COBOL-centric talent is a three-way collective responsibility held by the commercial sector, the technology sector, and academia. For its part, OpenText's investments in technology ensure incumbent and would-be COBOL developers receive every assistance possible.

Furthermore, hundreds of academic institutions have signed up to industry initiatives such as the OpenText™ COBOL Academic Program, which provides free access to the latest teaching tools for COBOL application development, adding to the hundreds of thousands of skilled COBOL developers in the market.

"Every other development language we've used over the years for specialized functions had to be replaced or discarded. COBOL has always endured and is unbelievably valuable for our business. Keeping pace in a professional sports market isn't easy, but using COBOL has enabled us to adapt to any hardware or software change and embrace new technology. We are entering our 30th year for our major league baseball software, and we still have an almost 90% market share."

MIKE PHILLIPS
Sydex Sports

Section 3. A Secure Future

The special ingredients that give COBOL its everlasting power

Past, Present, Future Perfect

Detailed histories exist of the COBOL language. In addition to the Wikipedia[15] entry, commentary from industry experts such as Bill Klein[16], original pioneers including Jean Sammet[17], and an entertaining study from practitioner Mike Madden[18], paint a colorful picture of the inception and lifespan of the language. They all point to 1959 as the date of the inception of the language (it was 1960 before any code was running). COBOL's success has been discussed before, of course (here's a 2009 discussion[19]).

Paraphrasing previous reviews, the argument goes that if the original design is sensible and well-considered, it stands a good chance at least of starting off well. So let's consider the factors that have helped it not only become popular, but also remain popular in 2019 and beyond.

> **"We continue to have employers tell us that COBOL is still a necessary language for their companies. They appreciate that we still offer a course in COBOL to help prepare students for jobs. There are few programming languages that can boast of being relevant and in use at age 60!"**
>
> **LISA M LANDGRAF**
> Computer Science & Software Engineering
> University of Wisconsin–Platteville

Designed to Succeed

COBOL is the original business language. That's what the B is for—Common *Business* Oriented Language. And "Business Oriented" suggests something important here. The name COBOL, and the well-defined computer language, arrived as a result of a specification framework created in September 1959[20]. The objective was to establish a means of enabling non-computer-literate professionals to communicate more effectively with computers, to support a growing need for computing services in government and industry. The framework included key requirements that "the language must be open-ended and capable of accepting change and amendment, that it should be problem-oriented and machine-independent, and that it should use simple English or pseudo-English and avoid symbolism as far as possible." With Grace Hopper[21] instrumental in the planning, the first incarnation, codenamed COBOL-60, took shape, within a year.

Staying Power

While a smart design is a good start point, staying power for decades is no easy thing. To help understand why COBOL has thrived, let's examine five core attributes any technology would need to ensure it had staying power. How might COBOL score against each of the following?

- **Innovation**—how can it stay contemporary to support changing technical needs?
- **History**—what support, reputation, marketplace and skilled user community can it rely on?
- **Portability**—how widely can the language prevail in an ever-evolving IT infrastructure?
- **Business-Centricity**—is it genuinely capable of supporting business-critical applications?
- **Readability**—who is going to learn it, use it, and maintain it, across the years?

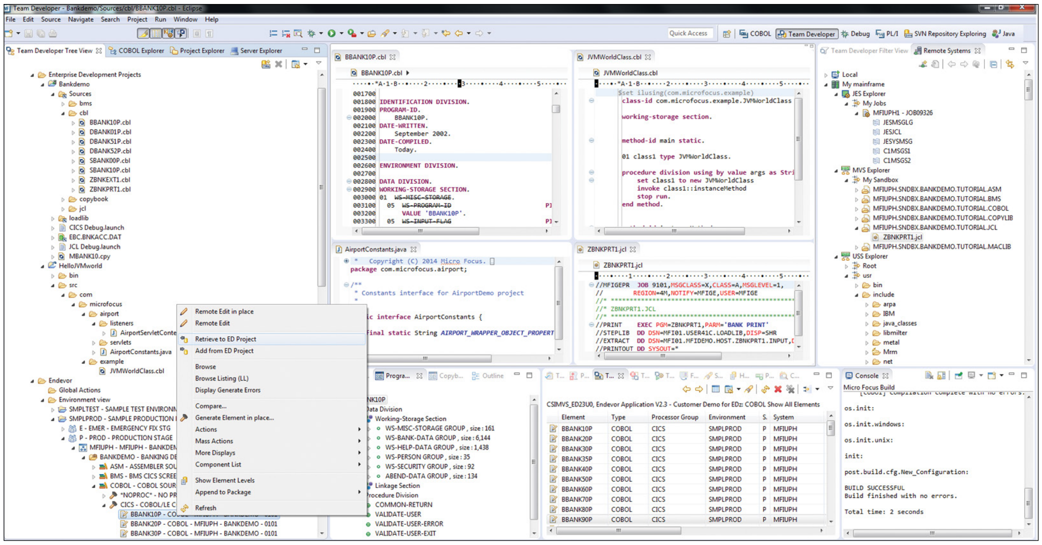
INNOVATION

Ensuring applications meet today's and tomorrow's requirements

Whatever your chosen programming language, it has to adapt to support the changing IT landscape. Tens of millions of dollars are invested annually to ensure COBOL remains contemporary. COBOL applications, regardless of deployment destination, can be maintained and enhanced using modern IDE frameworks (Eclipse or Visual Studio). COBOL is no different to any other language environment in that regard.

Additionally, COBOL applications from the past can simply recompile and run in the cloud, .NET and JVM, as well as more established operating environments such as mainframes, Linux, UNIX and Windows. More recently, exposing a COBOL application in a Docker container[22], for example, is possible in a few simple steps.

Furthermore, COBOL supports contemporary technology and integration with other language applications such as Java, C++ & C#. In addition, the growing use of SOA and Web Services, and the range of technology elements—such as REST/JSON, XML, WSDL, SOAP, HTML, ensure that application integration and connectivity is possible across the enterprise. Hooking into COBOL apps is a key need for new, digital systems, because that's where critical processing facilities and data reside. Again, COBOL has evolved to support a range of contemporary digital technology to ensure applications can interlock and integrate however needed.



```
BBANK10P.cbl
001700
001800 IDENTIFICATION DIVISION.
001900 PROGRAM-ID.
002000     BBANK10P.
002100 DATE-WRITTEN.
002200     September 2002.
002300 DATE-COMPILED.
002400     Today.
002500
002600 ENVIRONMENT DIVISION.
002700
002800 DATA DIVISION.
002900 WORKING-STORAGE SECTION.
003000 01  ~~WS-MISC-STORAGE~~.
003100   05  ~~WS-PROGRAM-ID~~                          PI
003200       VALUE 'BBANK10P'.
003300   05  ~~WS-INPUT-FLAG~~                          PI
```

```
JVMWorldClass.cbl
$set ilusing(com.microfocus.example)
 class-id com.microfocus.example.JVMWorldClass

 working-storage section.

 method-id main static.

 01 class1 type JVMWorldClass.

 procedure division using by value args as Stri
     set class1 to new JVMWorldClass
     invoke class1::instanceMethod
     stop run.
 end method.
```

```
AirportConstants.java
 *   Copyright (C) 2014 Micro Focus.
package com.microfocus.airport;

/**
 * Constants interface for AirportDemo project
 *
ic interface AirportConstants {

final static String AIRPORT_WRAPPER_OBJECT_PROPERT
```

```
ZBNKPRT1.jcl
//MFIGEPR  JOB 9101,MSGCLASS=X,CLASS=A,MSGLEVEL=1,
//         REGION=4M,NOTIFY=MFIGE,USER=MFIGE
//* ***********************************************
//* ZBNKPRT1.JCL
//* ***********************************************
//PRINT    EXEC PGM=ZBNKPRT1,PARM='BANK PRINT'
//STEPLIB  DD DSN=MFI01.USER41C.LOADLIB,DISP=SHR
//EXTRACT  DD DSN=MFI01.MFIDEMO.HOST.ZBNKPRT1.INPUT,D
//PRINTOUT DD SYSOUT=*
```

NK10P
Data Division
Working-Storage Section
WS-MISC-STORAGE GROUP , size : 161
WS-BANK-DATA GROUP , size : 6,144
WS-HELP-DATA GROUP , size : 1,438
WS-PERSON GROUP , size : 35
WS-SECURITY GROUP , size : 92
ABEND-DATA GROUP , size : 134
Linkage Section
Procedure Division
COMMON-RETURN
VALIDATE-USER
VALIDATE-USER-ERROR
VALIDATE-USER-EXIT

CSIMVS_ED23U0, Endevor Application V2.3 - Customer Demo for EDz: COBOL Show All Elements

Element	Type	Processor Group	Environment	S.	System
BBANK10P	COBOL	CICS	SMPLPROD	P	MFIUPH
BBANK20P	COBOL	CICS	SMPLPROD	P	MFIUPH
BBANK30P	COBOL	CICS	SMPLPROD	P	MFIUPH
BBANK35P	COBOL	CICS	SMPLPROD	P	MFIUPH
BBANK40P	COBOL	CICS	SMPLPROD	P	MFIUPH
BBANK50P	COBOL	CICS	SMPLPROD	P	MFIUPH
BBANK60P	COBOL	CICS	SMPLPROD	P	MFIUPH
BBANK70P	COBOL	CICS	SMPLPROD	P	MFIUPH
BBANK80P	COBOL	CICS	SMPLPROD	P	MFIUPH
BBANK90P	COBOL	CICS	SMPLPROD	P	MFIUPH

```
Micro Focus Build
     [cobol] Compilation complete with no errors.
os.init:

os.init.windows:

os.init.unix:

init:

post.build.cfg.New_Configuration:

BUILD SUCCESSFUL
Build finished with no errors.

Total time: 2 seconds
```

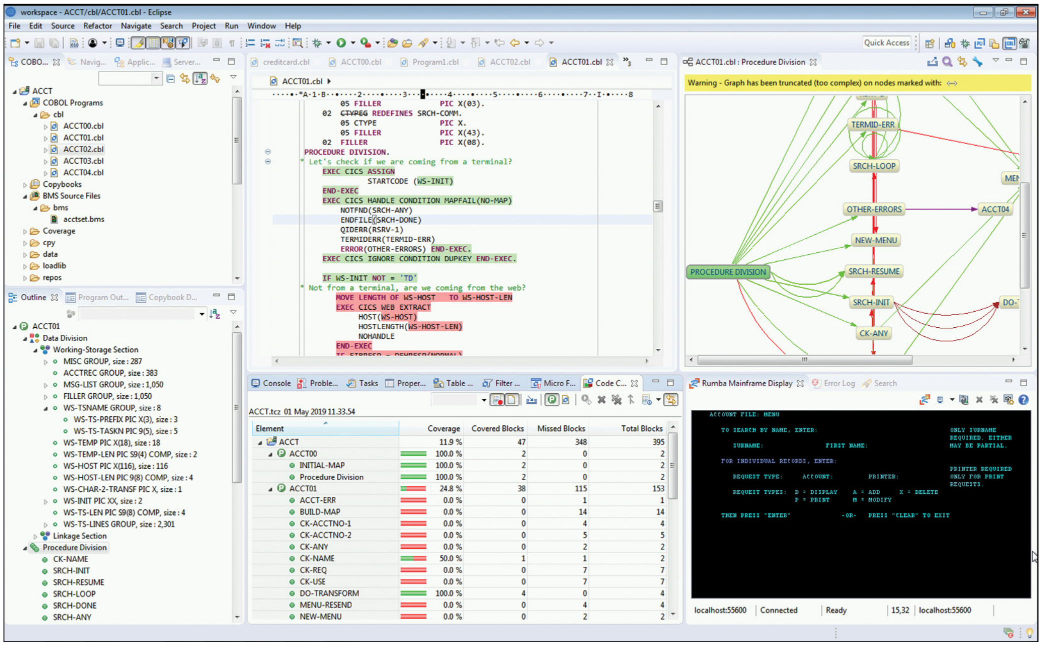
Figure 2. COBOL integrates with other languages and constructs as required

Nowadays developers are building managed code apps, cloud apps and containerized apps, as well as IBM mainframe (CICS, JCL, Db2, IMS) applications all within contemporary IDEs, using COBOL. There are remarkable contemporary use-cases for how COBOL applications can be enhanced and extended.

> **"Over the last 25 years, SYNTAX has helped many organizations modernize their COBOL applications, providing new capabilities and extending them to cutting edge technologies. We're amazed at the continuous value that this 60-year-old language has provided to many global organizations. This value is now easily expanded to new platforms, including the cloud, and supports through modern development tools, the next generation of COBOL developers."**
>
> **PANOS ZOTOS**
> SYNTAX Information Technology Inc

"COBOL has had a remarkable impact over the past six decades, but I believe we are only now realizing its full potential and that its future impact will be even greater. Thanks to Micro Focus (now part of OpenText™), COBOL is now a modern language, portable and fully object-oriented with an ability to integrate with both Java and .NET frameworks. This revolutionary language … enables a new era of innovation for developers supporting existing business applications, process and data. In my view, nothing else compares to this remarkable language designed for today's business needs and built to last."

BOB ENGLAND
CTO
England Technical Services


workspace - ACCT/cbl/ACCT01.cbl - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
COBO... Navig... Applic... Server...
ACCT
COBOL Programs
cbl
ACCT00.cbl
ACCT01.cbl
ACCT02.cbl
ACCT03.cbl
ACCT04.cbl
Copybooks
BMS Source Files
bms
acctset.bms
Coverage
cpy
data
loadlib
repos
Outline Program Out... Copybook D...
ACCT01
Data Division
Working-Storage Section
MISC GROUP, size : 287
ACCTREC GROUP, size : 383
MSG-LIST GROUP, size : 1,050
FILLER GROUP, size : 1,050
WS-TSNAME GROUP, size : 8
WS-TS-PREFIX PIC X(3), size : 3
WS-TS-TASKN PIC 9(5), size : 5
WS-TEMP PIC X(18), size : 18
WS-TEMP-LEN PIC S9(4) COMP, size : 2
WS-HOST PIC X(116), size : 116
WS-HOST-LEN PIC S9(8) COMP, size : 4
WS-CHAR-2-TRANSF PIC X, size : 1
WS-INIT PIC XX, size : 2
WS-TS-LEN PIC S9(8) COMP, size : 4
WS-TS-LINES GROUP, size : 2,301
Linkage Section
Procedure Division
CK-NAME
SRCH-INIT
SRCH-RESUME
SRCH-LOOP
SRCH-DONE
SRCH-ANY
creditcard.cbl ACCT00.cbl Program1.cbl ACCT02.cbl ACCT01.cbl
ACCT01.cbl : Procedure Division
Warning - Graph has been truncated (too complex) on nodes marked with: ⇔
TERMID-ERR
SRCH-LOOP
MEN
OTHER-ERRORS
ACCT04
NEW-MENU
PROCEDURE DIVISION
SRCH-RESUME
SRCH-INIT
DO-
CK-ANY
Console Proble... Tasks Proper... Table ... Filter ... Micro F... Code C...
Rumba Mainframe Display Error Log Search

ACCT01.cbl

```
      ·····•·*A·1·B··•·····2·····•·····3·····•·····4·····•·····5·····•·····6·····•·····7··I··•·····8
                05 FILLER               PIC X(03).
             02 CTYPEG REDEFINES SRCH-COMM.
                05 CTYPE                PIC X.
                05 FILLER               PIC X(43).
             02 FILLER                  PIC X(08).
         PROCEDURE DIVISION.
        * Let's check if we are coming from a terminal?
             EXEC CICS ASSIGN
                       STARTCODE (WS-INIT)
             END-EXEC
             EXEC CICS HANDLE CONDITION MAPFAIL(NO-MAP)
                NOTFND(SRCH-ANY)
                ENDFILE(SRCH-DONE)
                QIDERR(RSRV-1)
                TERMIDERR(TERMID-ERR)
                ERROR(OTHER-ERRORS) END-EXEC.
             EXEC CICS IGNORE CONDITION DUPKEY END-EXEC.

             IF WS-INIT NOT = 'TD'
        * Not from a terminal, are we coming from the web?
                MOVE LENGTH OF WS-HOST   TO WS-HOST-LEN
                EXEC CICS WEB EXTRACT
                     HOST(WS-HOST)
                     HOSTLENGTH(WS-HOST-LEN)
                     NOHANDLE
                END-EXEC
```

ACCT.tcz 01 May 2019 11.33.54

Element	Coverage	Covered Blocks	Missed Blocks	Total Blocks
ACCT	11.9 %	47	348	395
ACCT00	100.0 %	2	0	2
INITIAL-MAP	100.0 %	2	0	2
Procedure Division	100.0 %	2	0	2
ACCT01	24.8 %	38	115	153
ACCT-ERR	0.0 %	0	1	1
BUILD-MAP	0.0 %	0	14	14
CK-ACCTNO-1	0.0 %	0	4	4
CK-ACCTNO-2	0.0 %	0	5	5
CK-ANY	0.0 %	0	2	2
CK-NAME	50.0 %	1	1	2
CK-REQ	0.0 %	0	7	7
CK-USE	0.0 %	0	7	7
DO-TRANSFORM	100.0 %	4	0	4
MENU-RESEND	0.0 %	0	4	4
NEW-MENU	0.0 %	0	2	2

```
  ACCOUNT FILE: MENU

     TO SEARCH BY NAME, ENTER:                                   ONLY SURNAME
                                                                 REQUIRED. EITHER
        SURNAME:                FIRST NAME:                      MAY BE PARTIAL.

     FOR INDIVIDUAL RECORDS, ENTER:
                                                                 PRINTER REQUIRED
        REQUEST TYPE:     ACCOUNT:          PRINTER:             ONLY FOR PRINT
                                                                 REQUESTS.
        REQUEST TYPES:  D = DISPLAY    A = ADD     X = DELETE
                        P = PRINT      M = MODIFY

     THEN PRESS "ENTER"           -OR-   PRESS "CLEAR" TO EXIT
```

localhost:55600 | Connected | Ready | 15,32 | localhost:55600

Figure 3. OpenText development tooling supports creation of mainframe and distributed COBOL applications

Finally, COBOL continues to evolve to take advantage of emerging technology: In 1974, 1985, 2002 and 2013, the COBOL language standard was updated. OpenText has participated in the various governing standards bodies, and is also one of the key pioneers of the new standard from an innovation and compliance perspective.

HISTORY

Six decades of heritage, billions of lines of value, hundreds of thousands of practitioners

New application development rarely starts from a blank sheet of paper. Innovations often arise from delivering business applications through new channels. Using the extensive business logic built into existing COBOL systems has made a lot of business sense for a long time.

- **Reusability**—COBOL's highly reusable nature is why COBOL permeates the enterprise. Why write new if the business function already exists?
- **Accessibility**—Alternative development languages can rapidly access COBOL value using native semantics and data types
- **Compatibility**—Another rung on the ladder to ubiquity is the concept of Backwards Compatibility any COBOL program, anywhere, which conforms to the standard(s), will compile with the latest COBOL product. Supported by the vendors, this guaranteed forward path serves to keep applications current

and provides a low-risk environment for systems development. This contrasts starkly to other technology options, where the code was good for a couple of years but had to be rewritten because a compiler release had changed.

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. While it is not easy to assess the relative value or popularity of computing languages against each other, one respected industry measurement, the TIOBE index[23], does just that. TIOBE's index of language popularity (the index measures and ranks hundreds of languages) shows COBOL in the 20–30 range during 2019. More significantly, the same rating shows COBOL as present in the top 30 since ratings began in 1989 , one of only 3 languages that can make such a claim over that period of time (the others are C and C++. Java was not ranked until later in the 1990s).

Such destiny appears to have been foretold: the following is valid COBOL syntax

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

PORTABILITY

Designed to run anywhere, unchanged

Choice is a defining characteristic of the IT industry. Whatever the question—there are multiple answers. Whatever the challenge—there are multiple options. This is especially true for platforms. Most enterprises sport disparate, heterogeneous Hybrid IT environments. Increases in processor power and hardware commoditization present IT buyers with choices when it comes to platform selection: Windows, Linux, UNIX, PC, Desktops or VMs, Mainframe or non-Mainframe, and of course all this can be on or off-premise, hosted in a Data Center or Cloud.

The blurring of demarcation between platform choices now means that software vendors must make their applications available on a wider range of platforms. Customers will, of course, choose an application based on its functionality, but the decision will also consider the breadth of platforms supported, relative ownership costs, user requirements, skills profiles, supply-chain policy, etc.

Modern COBOL technology enables today's developer to analyze, develop, debug, test and deploy their applications across an array of platforms. The integrated development environments allow for instantaneous edit/debug cycles, feature-rich developer tooling, and running the same portable code under new industry-leading frameworks such as .NET and Eclipse, containers and Cloud.

> **"Throughout our products' evolution we have found that COBOL's strengths are its portability, applicability and a very natural language-like structure. Its ease of programming, certainly in English, is second to none and can allow novice programmers to dive right in. I believe COBOL's longevity will be assured, thanks both to Grace Hopper, and to the continued investment and development by Micro Focus (now part of OpenText™)…here's to the next 60 years!"**
>
> **VANESSA WHITE**
> Product Owner
> Advanced

*https://freedomafterthesharks.com/2016/06/27/exactly-what-is-cobol-and-why-is-cobol-still-a-widely-used-language-in-it/

> **"Most people don't realize their daily routines depend on 60-year-old technology. Whether it's using an ATM, booking travel or filing an insurance claim, we all interact with COBOL-based systems in some way each day, and will continue to do so for the foreseeable future. This is an undeniable testament to the longevity and continued relevance of COBOL, and I believe it will continue to run mission-critical applications in the cloud for another 60 years."**
>
> **CRAIG MARBLE**
> Astadia

COBOL deploys across all leading enterprise platforms today, as it has always done. While Java is often labelled the language of portability, it falls short of the breadth achieved by COBOL.

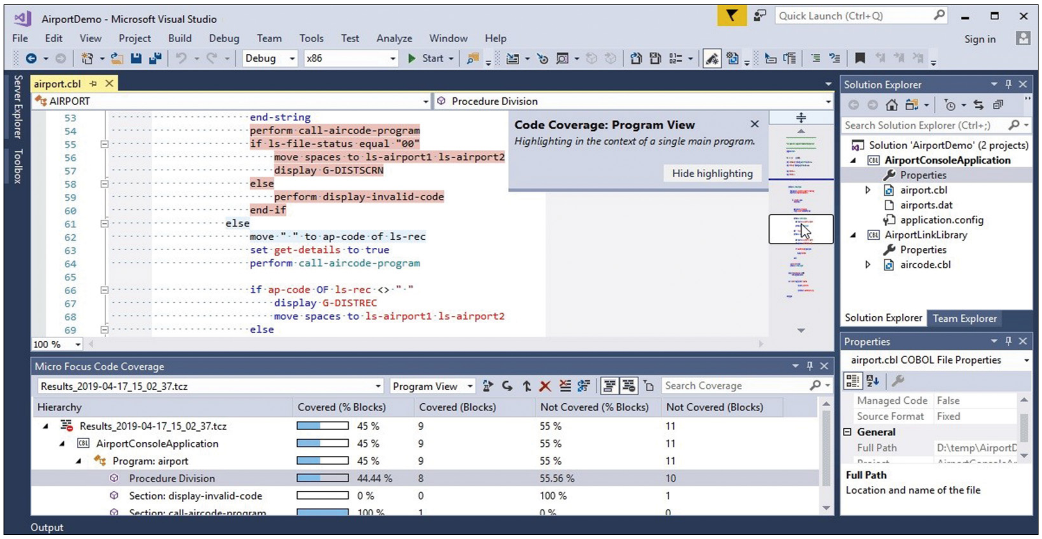
BUSINESS-CENTRICITY

Engineered for building great business applications

Core enterprise systems need strong and reliable IT infrastructures that offer robustness and validity, strong data manipulation, accuracy, speed and accessibility. In short, something fit for business needs, and designed with scale in mind. OpenText leverages COBOL strengths as a business critical programming language in the following ways:

- **Robustness and validity—apps are low-risk**
  COBOL's type-rich language allows data to be described accurately with explicit scope and limits. This richness means you can meet your corporate coding standards, ensuring consistency and accuracy across your organization and third parties, including partners and industry specific compliance requirements.
- **Numeric arithmetic accuracy—always right, all the time**
  As a business-centric language COBOL delivers arithmetic accuracy to 38 decimal digits—that's more than almost every other language. The accuracy of calculations in many of the world's largest systems cannot be a point of compromise, which is why many of the world's financial powerhouses rely on COBOL.
- **Strong data manipulation—protect data integrity**
  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. This uniquely enables you to SORT and filter within COBOL without having to engage another tool or any extra steps, and is much faster than having to handle this outside the language.

- **Performance—execute at the speed of business**
  Speed of application execution is a key criterion for core business applications, which need horsepower to deliver both computational speed and batch support. COBOL applications can be optimized for specific hardware and platforms, increasing performance and throughput. OpenText's optimizing COBOL compiler exploits target platform technology to deliver optimal performance. OpenText's optimization capability has helped organizations improve application performance significantly, so much so in fact that COBOL is often regarded as the gold standard benchmark for application performance.
- **Accessibility—available wherever needed**
  Programming languages don't exist in isolation. They only deliver value in the hands of the coders, who in turn are only as good as their ability to align the application to a business need. To make a valuable business contribution, the language should also combine a state of the art environment for building robust apps, including latest features (auto-correct, rapid code/debug, UI builder) and be built on the latest frameworks (OpenText™ Visual Studio and OpenText™ Eclipse) and, crucially, be available for developers regardless of the intended deployment platform.



```
                    end-string
                    perform call-aircode-program
                    if ls-file-status equal "00"
                        move spaces to ls-airport1 ls-airport2
                        display G-DISTSCRN
                    else
                        perform display-invalid-code
                    end-if
                else
                    move " " to ap-code of ls-rec
                    set get-details to true
                    perform call-aircode-program

                    if ap-code OF ls-rec <> " "
                        display G-DISTREC
                        move spaces to ls-airport1 ls-airport2
                    else
```

Code Coverage: Program View
Highlighting in the context of a single main program.

Micro Focus Code Coverage

Results_2019-04-17_15_02_37.tcz
Program View
Search Coverage

Hierarchy	Covered (% Blocks)	Covered (Blocks)	Not Covered (% Blocks)	Not Covered (Blocks)
Results_2019-04-17_15_02_37.tcz	45 %	9	55 %	11
AirportConsoleApplication	45 %	9	55 %	11
Program: airport	45 %	9	55 %	11
Procedure Division	44.44 %	8	55.56 %	10
Section: display-invalid-code	0 %	0	100 %	1
Section: call-aircode-program	100 %	1	0 %	0

airport.cbl COBOL File Properties

Managed Code	False
Source Format	Fixed
General	
Full Path	D:\temp\AirportD

Full Path
Location and name of the file

Figure 4. Feature-rich IDEs support rapid application development of COBOL systems for both mainframe and distributed platform deployment

> **"COBOL is still quietly running the world's most critical business applications. With advanced optimization technology and modern development tools, companies are able to modernize COBOL applications to keep up with evolving business, performance, and digital transformation needs. COBOL is by far the best language for business. It will remain strategic and maintain its reputation of being a fast, robust and future-proof programming language."**
>
> **ROLAND KOO**
> Program Director, Offering Management and Strategy, Enterprise Products and Compilers on Z, IBM.

"After 60 years, COBOL is still the language of business. Designed for maximum efficiency and ease of use, this proven programming technology delivers unrivaled data processing speed and precision at enterprise scale. With millions of global transactions processed every second, COBOL delivers adaptability and performance, outpacing many of its modern rivals. While some would question the … continued use COBOL, I see a language that's quietly powered six decades of economic growth and a technology with a very bright future ahead."

TOM ROSS
Senior Software Engineer
IBM

READABILITY

Ease of learning, reading and writing enables you to focus on business

COBOL is simple to understand and can be learned quickly. This contrasts with many programming languages where, even with the skills to write it, the code is hard to understand. COBOL is structured in terms of its layout, and uses active English-derived constructs (the syntax ADD means add, EQUALS is equals, IF is if, and so on). The reader knows at a glance what the code is trying to achieve, a huge advantage.

- As anyone can write it, it becomes possible to create low-cost high-availability resource pools to construct applications. As anyone can read it the downstream benefits are equally significant. First, it means the resource pool available to work on COBOL systems is conceptually unlimited. There is no barrier to entry for future COBOL programming skills. This means a lot in terms of strategic planning and investment.
- Second, if anyone can read it anyone can maintain it. A fresh generation of programmers can code COBOL and also maintain existing COBOL applications they hadn't originally written. Java and C# teams can review the COBOL back-end to their new front-end code, using the same IDE. Non-developers can follow the program flow, QA staff can assist with code walkthroughs and debugging work, and so on. Additional tooling to assist with application understanding and impact analysis provides a faster on-ramp for those learning the syntax for the first time.
- Third, the high legibility of COBOL avoids a major and common pitfall—namely that coders will simply rewrite something they do not understand in their preferred language. COBOL typically passes the comprehension test, in a way that many other languages may fall out of vogue over time. "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."

Conclusion: Shine Like a Diamond

A year is long time in technology. Six decades of commercial value is therefore a phenomenal achievement. COBOL pre-dates Microsoft, SAP, Oracle, UNIX, Linux, Windows, Java, the internet and even the IBM mainframe. COBOL has seen it all and taken it in its stride, by continuing to modernize itself in support of an industry that is doing the same.

"Modernization can and needs to take many forms. Micro Focus (now part of OpenText™) represents that kind of flexibility with a portfolio of solutions that allow customers to implement the most valuable approach to modernization based on their specific needs," Modernization: A Flexible Approach to Digital Transformation, Peter Rutten, IDC, 2018.

Looking ahead, as modernization use cases evolve and twist to reflect the ever-changing digital landscape, COBOL will follow the path of technology innovation. Investments to support more diverse use-cases in contemporary technology, such as cloud native data, scale-out execution performance, serverless computing, and containerization are already underway. If the innovations COBOL supports through 2019 read like a list of the industry's best ideas of a generation, the next chapter should expect to see similar breakthroughs that would probably evade even the wildest speculation.

COBOL has stood the test of time because it was designed for business, and has evolved to support the changing world around it. As the telephone and motor car metaphorically demonstrate, great ideas need to evolve to adapt to their environment. Genuinely smart technology does the same.

In IT, the pressure to deliver more, faster, is unprecedented. What emerges from such unyielding pressure is the occasional gem, perfected over time, and hugely valuable. As Kissinger once remarked, "A diamond is a chunk of coal that did well under pressure."

Shine like a diamond, COBOL; happy anniversary.

"COBOL plays an important role in the digitization of core business systems. For IT leaders, focused on key transformation initiatives, re-use of COBOL application logic, from mainframe to cloud, delivers new business flexibility and value. Now 60 years on, modern COBOL delivers even greater possibilities for the developer supporting the digital age."

BOB ELLSWORTH
Director Strategic Workloads
Microsoft

"I first learned to program in 1970, almost 50 years ago. At that time, COBOL was the new kid on the block. Now 60 years old, COBOL has not only stood the test of time but is still going strong."

LAURIE WALLMARK
Author of many popular children's books including *Grace Hopper: Queen of Computer Code*

"COBOL not only offers the most flexible, multi-platform . . . language on the market but also now delivers new integration with modern technologies including Visual Studio, Eclipse, agile tools and DevOps practices; ensuring a promising career path for new programmers and a bright future for the mission-critical business applications it supports globally."

HAL PETERS
Pinebrook Consulting

References

1. www.reference.com/world-view/said-only-thing-constant-change-d50c0532e714e12b
2. www.codingame.com/blog/best-programming-language-learn-2019/
3. www.techworm.net/2018/02/popular-programming-languages-2018-according-tiobe-pypl.html
4. https://en.wikipedia.org/wiki/Distributed_ledger
5. www.aei.org/publication/fortune-500-firms-in-1955-vs-2014-89-are-gone-and-were-all-better-off-because-of-that-dynamic-creative-destruction/
6. www.zdnet.com/article/worldwide-cost-of-it-failure-revisited-3-trillion/
7. www.gartner.com/document/3872986
8. www.microfocus.com/media/infographic/the_future_of_cobol_applications_2017_survey_highlights.pdf
9. www.marketsandmarkets.com/PressReleases/application-modernization-services.asp
10. www.microfocus.com/idc-modernization
11. www.microfocus.com/en-us/products/enterprise-analyzer/overview
12. www.microfocus.com/en-us/products/cobol-analyzer/overview
13. www.microfocus.com/en-us/products/enterprise-developer/overview
14. www.microfocus.com/en-us/products/visual-cobol/overview
15. https://en.wikipedia.org/wiki/cobol
16. https://web.archive.org/web/20140107192608/http:/home.comcast.net/~wmklein/DOX/History.pdf
17. https://en.wikipedia.org/wiki/Jean_E._Sammet
17. www.legacyit.co.uk/happy-birthday-dear-cobol-part-one/
19. www.theguardian.com/technology/2009/apr/09/cobol-internet-programming
20. https://web.archive.org/web/20140107192608/http:/home.comcast.net/~wmklein/DOX/History.pdf
21. https://blog.microfocus.com/amazing-grace-hopper/
22. www.microfocus.com/campaign/download/visualcobol4/
23. www.tiobe.com/tiobe-index/

Connect with Us

OpenText CEO Mark Barrenechea’s blog

opentext™