COBOL and the Business Programming Paradigm

by Jonathan Sayles, IBM

"Why COBOL"

Have you gone to a Technology conference lately? If you did, you probably couldn't miss listening to some sarcastic remark about the COBOL language, COBOL development and even COBOL developers. Yes, it's 1996, and the anti-COBOL dogmatists are still on the warpath, with their "new **AS** improved" (as opposed to the more accurate, "**new as unproven**") line of rhetoric on how to reform I/S.

Only this time they're taking no prisoners. Their pitch is that COBOL is a computing language whose time has come and (long since) gone. And that I/S's only salvation from COBOL lies in rewriting production applications using modern (**alternative**) technology. Managers, directors and vice presidents in attendance should be alert to the implications of this message: *"COBOL may be what's always - and is currently - keeping your data center operational, but you're better off staking your personal - and the company's - technical and financial fortunes on technologies with little-to-no track record in the production world."* For all the hype you'd suppose that 95% of the production business applications in the Fortune 500 were written in alternative technologies and only 5% were written in COBOL!

Adding insult to injury

Not content to stop at burying the language, many *alternative-technology-proponents* feel compelled to demean COBOL developers, stooping to imply that they are "dim-wits" all of whom should be replaced by recent college-graduates armed with the latest trendy software and silver bullet products. Some have even gone so far as to allege that the reason for past silver bullet failures can be blamed entirely on those dumb-old, incorrigible COBOL programmers - who just can't help but get in the way of *progress* (remember "new **AS** improved"). Few have honesty and candor to admit that silver bullet technologies fail because they lack the bandwidth necessary to cope with the complexities intrinsic to production business requirements, production business capacity and scale.

Dave Babson, president of Burl Technologies sums it up nicely; "The new technology providers bash COBOL because they must create a **throw away - build from scratch mentality** in order for their products to succeed." And this new wave of anti-COBOL dogma might seem so much tripe were it not so pervasive and unchallenged by technicians unwilling to be politically incorrect. So, others can bury Caesar, I'll praise him.

Why is it that COBOL is still an excellent **fit** with the needs of large-scale, complex, long-lived business computing and application development?

COBOL and the Business Application Paradigm

The first COBOL committee - a group of independent language experts and I/S technicians from all walks of life (many commercial-industry (finance, manufacturing, retail,etc.), but including

computer software vendors, government, and university - academia!) was convened to design a business oriented computing language that was to become a standard throughout the commercial data processing world. Besides taking into account what computers were capable of - at the time, and what business at the time needed from a general-purpose data processing language, the group took into serious consideration that there are fundamental, critical and non-negotiable realities of business application development. Let us call these realities the **"Business Application Paradigm."** They are listed below in figure one.

- **Business Applications are large, very large and very, very large**

- **Business Applications are magnitudes of order more complex than dog&pony show demos**
- **Business Applications are long-lived (10, 20 and 30 year-old applications are common)**
- **Business Applications are dynamic - During the time business applications spend in production they grow in complexity, and evolve well beyond the original design and specification requirements**
- **Business Applications are critical - Applications must be stable at high transaction and data access volume and capacity (often measured in the millions/day). If applications ABEND they must be fix-able, and reStart-able under duress**
- **Business Computing Architectures evolve and change**
- **Business comes first - Technology for technologies sake is not good business**
- **Business projects will be understaffed and overworked**
- **Millions of business programmers are needed to meet worldwide business application development requirements**
- **Business project teams change - individuals responsible for writing applications rarely maintain and support them in production**
- **I/S departments are unique - no two shops employ the same methods, standards and practices in developing, maintaining and supporting production code**

Figure 1. The realities of the Business Application Paradigm ***- yesterday, today and tomorrow***

After much analysis, research and discussion, the independent COBOL committee came up with the first COBOL language standard. COBOL was designed to meet both the technical requirements for business processing at the time, and the critical requirements of the Business Application Paradigm. The committee did an excellent job of reconciling the contradictory requirements inherent in Figure 1. Through their intelligence, independence, foresight and hard work, they created a standard, pragmatic general-purpose business-oriented computing language, which over time has kept production systems - and the businesses they support - operational.

COBOL, as a business computing language is the only **single** language that meets the requirements of figure 1. Other languages may excel at one or a few of the elements in figure 1, but COBOL alone is:

Self-documenting

- COBOL is an English, readable, self-documenting programming language. Large, complex, and long-lived application development projects benefit from COBOL's self-documenting syntax and semantics. Most importantly - **maintenance** and **production support-related tasks** benefit, particularly when done by I/S technicians who are **not** the original application authors.

Simple

- COBOL encourages a simple, straight-forward programming style. Programmers develop complex applications combining simple procedural constructs, instead of writing impenetrable, terse code. This facilitates maintainable, production-enabled programs for large, long-lived, dynamic, complex applications.

NonProprietary-(portable)

- COBOL is far and away the most portable language. Because the independent ANSI-COBOL committee legislates non-vendor-specific syntax and semantic standards, COBOL applications can be developed, ported, down-sized, upsized, rehosted, reused and rightsized to every operating system on every hardware platform - PC, Network, mid-range, mainframe, Windows, OS/2, every flavor of Unix, DOS, AS/400, VSE, VMS, VM, MVS, you name it, COBOL's been there, done that.

Efficient

- COBOL has a 20 year history of **optimizing** compiler technology, which by now is sophisticated to the point of object code reengineering for greatest run-time efficiency specific to given hardware platforms.

Scalable

- Through COBOL's availability on all common I/S run-time environments, and through NonProprietary CALL USING constructs COBOL applications can scale up with loose or tight coupling - and even O-O message binding with ease. Complex business data processing activity which is supported by huge COBOL applications (millions of lines of code) is common and maintained through this simple, scalable construct.

Universal

- Well over 1,500,000 I/S technicians and programmers worldwide are **expert** in COBOL. COBOL solutions do not require hard-to-find, expensive consultants to develop, maintain

and support them (although lately, with all the fuss and attention paid to non-COBOL technologies, want-ads for COBOL programmers are dramatically on the rise!). Through the efforts of the ANSI committee COBOL programs written in Germany run on mainframes from Deli to Santa Cruz to Manhattan. The COBOL language is a constant from shop-to-shop, from project-to-project.

Open

- COBOL integrates and can converse with all other 3GLs (C, PL/I Fortran, etc.) all 4GLs (CA-Eztrieve, FOCUS, Visual Basic, PowerBuilder, Delphi), assembler languages (ALC), pure and hybrid O-O languages (Smalltalk, C++...), API-based technologies (Windows SDK, DCE...), relational (aka Client/Server) DBMSs, non-relational DBMSs (IMS, CA-IDMS, Total), object DBMSs, and all pointer-based and sequential file systems.

Complete

- In spite of its simplicity and in addition to its business orientation COBOL is "computationally complete" - and can be used in applications that run the gamut from simple batch reporting to complex transaction and windowed systems across all industries and business and computing requirements.

Mature

- COBOL has thousands of 3rd-Party products from hundreds of companies with upwards of thirty years of support for the COBOL market. Hundreds of products exist in the critical areas of application testing, debugging, application analysis, maintenance, production support and code reuse. New COBOL compilers and development tools are announced every quarter.

Reliable and Proven

- Perhaps most significantly COBOL is used in more production systems and has a more proven and reliable production history than all other business languages combined.

Why COBOL?

Figure 2 describes the above COBOL attributes positioned against the realities and requirements of the Business Application Paradigm. As figure 2 makes apparent, COBOL evolved from the business development paradigm, and as a single language is uniquely suited for business application development, maintenance and production support. This is not to say that COBOL is all you need to create contemporary (particularly GUI) applications. Nor is it say that COBOL is even best for every class and type of business application. But if you are developing large, complex, long-lived production systems, that will be maintained by a heterogenous group of

developers COBOL is the one constant that can be depended on to protect your investment in business application development in an age where intemperate change in I/S technology accelerates towards warp speed.

Is it any wonder that COBOL's death, while anxiously anticipated by vendors wishing to replace the language with their own proprietary wares, is "greatly exaggerated"? Is it any wonder that COBOL is still the "language of choice" for Enterprise application development? **Gartner Group August 1995** reports that COBOL is used in over 65% of **new** application development! The only wonder is that COBOL doesn't dominate the I/S tabloids and conference seminars. But then, what works has never been as interesting as what's fashionable. And fashionable is conspicuously absent from the business application paradigm.

---------- COBOL Language Elements and Properties ----------

Business Application Paradigm Realities and Requirements	Self-Documenting	Simple	Portable	Efficient	Scalable	Universal	Open	Complete
Large Applications	x	x	x	x	x	x	x	x
Complex Applications	x	x	x	x	x	x	x	x
Long-lived Applications	x	x	x	x	x	x	x	x
Dynamic Applications	x	x	x	x	x	x	x	x
Critical Applications	x	x	x	x	x	x	x	x
Evolving Architectures	x	x	x	x	x	x	x	x
Business-orientation	x	x	x	x	x	x	x	x
Project understaffing	x	x	x	x	x	x	x	x
Bell-shaped curve of talent	x	x	x	x	x	x	x	x
Every shop is unique	x	x	x	x	x	x	x	x
Maintenance/production supp.	x	x	x	x	x	x	x	x

Figure 2, How the COBOL language properties support the Business Computing Paradigm

Jonathan Sayles, Business Application Developer, IBM/Rational Software, Mr. Sayles has published 16 books and ~100 articles on topics such as relational database, Enterprise modernization, Rational Software products, client/server development, application development workbenches, Visual Basic, PowerBuilder, DB2, Oracle, and SQL. He has been editor of the DB2 Journal, and the Relational Database Journal. Mr. Sayles has spoken at IBM, Rational, Fortune 100 company, Powersoft, VBits, IBM, Micro Focus, and DBExpo international conferences, as well as hundreds of regional conferences. As a technical consultant, Mr. Sayles has worked in over 80 of the Fortune 1000 shops in the last 10 years.