

The new face of COBOLSM

ACUCORP[®]

The Future of COBOL—An Acucorp View

by Gerold Ekström

January 27, 2003

I find it quite interesting to try to predict COBOL's future, especially because I personally heard the "death of COBOL" being declared in the second half of the 1970s. This happened in the Concert House of Stockholm, Sweden. The death sentence was issued by a representative for a mainframe-based 4GL. I, being young at the time, was quite impressed by the speaker's words, but as I was leaving the building a considerably older IT manager turned to me and said,

"Trust me, we will be programming in COBOL in the year 2020!"

In the late 80s, I was invited to give a presentation about COBOL at a computer conference in the same city. Afterwards, I winced as they thanked me for representing "the views of the older generation." It will probably come as no surprise to you that many of the "younger" languages feted at that conference are no longer in use today.

Although repeated "death sentences" have been issued for COBOL, and although there have been numerous attempts to bring about its demise, the language is still flourishing. I personally believe that my IT manager from the 70s was right: COBOL will still be used in the year of 2020. But probably not in the way that he imagined it!

As a COBOL vendor, I am excited about the opportunities that lie ahead for COBOL applications and the companies they serve. I think COBOL is well positioned for the future. I will explain why I think so, and how I believe Acucorp can help corporations mine the business rules embodied in their COBOL applications and use them to interoperate with all manner of new technologies—all without upheaval.

Why COBOL Thrives

As many COBOL programmers do, my manager in the 70s genuinely liked the language and appreciated its breadth, functionality, and English-like syntax.

But COBOL has always been controversial. Why is that so? Writing a program in COBOL takes more time than in most other programming languages. Instead of writing computer-oriented code ($A = B$), the developer uses natural language statements (MOVE B TO A). This takes time and may seem unnecessary to the programmer. But this type

of code has at least two significant business advantages that have contributed to its popularity.

First, employees who are solving a business problem can focus their attention on the solution rather than on the translation into code. COBOL's natural language statements mean that there is less focus on programming and more on business savvy.

Second, COBOL's English-like statements allow a maintenance programmer or other co-worker to read and understand code developed by another person. The COBOL itself remains meaningful to a business person long after code was written.

Another factor contributing to COBOL's continued existence is the structure of the language itself. The inventors of COBOL had the foresight to design an open-ended language structure that remains fundamentally the same after more than 40 years. With extensions to COBOL, you can adjust the language to accommodate almost any technological advance. Later in this paper, I will explore some of the ways that COBOL extensions interoperate with the newest technologies.

I think it's important to emphasize that the tremendous value inherent in millions of lines of COBOL code lies not only in the technology—as good as it is—but primarily in the business rules embodied in the code. The code encapsulates valuable knowledge that lies at the heart of the corporation. Formerly manual business practices and routines have been transformed into automated processes, enhanced, modified, and refined over and over again, to become applications that do their jobs effectively and at great benefit to the organizations they serve.

In many companies, there may no longer be an employee who could do the same business functions manually—or even describe how they should be done. The challenge businesses face is to access this encoded business logic, encapsulate it, and integrate it with the latest technologies in a way that presents a low risk to the operation of business.

Why an Opportunity Exists

As popular as it's been, we know that the outlook for COBOL isn't all clear sailing.

Older COBOL applications can no longer be called fashionable. COBOL programmers are aging along with their code, and the youngsters are generally not too keen to learn COBOL.

Locating good COBOL training programs is getting harder. And many COBOL applications run on hardware that has been declared obsolete—or soon will be.

A number of attempts have been made to overcome the lack of organization in COBOL applications. Structured programming techniques like Jackson Structured Programming (JSP) and various code generators such as Synon have been developed and embraced

over the years. Some of these tools were very successful, but their adoption was not universal, and the result has been that some COBOL applications are well structured and well documented, but many are monolithic and ill-documented, and thus much more difficult to maintain.

All of the above seem to be fairly good reasons for throwing out the applications and replacing them with new ones developed in newer languages that people like to learn.

So why are we still discussing COBOL's future? I think it is largely because of the sheer volume of well-functioning business routines hidden in the code. Financial institutions, insurance companies, manufacturers, wholesalers, distributors, transportation, retail chains, and dozens of other industries rely on the smooth operation of their business processes. These companies are disinclined to disturb hundreds of thousands of lines of COBOL code that have worked so well for so long.

Certainly cost is also a factor. When the economy is tight, as it is now, businesses are extremely cautious about spending money that doesn't have a high, immediate payoff.

I think the bigger risk is the difficulty of understanding what these applications were designed to do, and then extracting that critical logic, so that it can be re-engineered for use with newer technologies. Certainly the difficulty of extracting the business logic looms large for monolithic applications that failed to embrace structured methodologies.

This presents a very real dilemma for security conscious (yet forward-thinking) businesses. How can a business have the best from both worlds: the trusted business rules encapsulated in their COBOL code, and the power and appeal of Java, C#, Visual Basic, Web Services, .NET, and an ever more dazzling array of connectivity tools? Assuming management believes that significant benefits can accrue from the newer languages and platforms (or that their IT staff firmly believes that newer is better), what about that COBOL code? Rewrite it? Replace it? Can it possibly be saved?

Rewriting all of a company's programs in a new language is an option, but it would be an expensive and risky proposition.

According to Capers Jones, (*Software Assessments, Benchmarks, and Best Practices*), rewriting a 10,000-function-point application in a new programming language is similar to original development, involving a long design, code, test, and implementation cycle.

His research indicates that:

- at least 50% of such projects run over budget.
- nearly one-quarter of them are behind schedule.
- more than 25% of them are rescheduled or cancelled.
- the final system has less than one-half of the planned features.

“We’re going to replace this software” often doesn’t happen, simply because a company doesn’t have the resources to do the job. Managers don’t have time to do a proper analysis of what the current application is doing. They don’t have time to uncover all of the special cases that were added as the application aged.

On top of all this, the economy is tight. Businesses will be facing this economic situation for a number of years to come, and so will COBOL application vendors.

If we can show companies how to extract and preserve the business logic encoded in their COBOL, so that it continues to work for them just as it always has—if we can show them a way to keep the COBOL code and still embrace the sizzling new technologies—then we have enabled COBOL to go on living and thriving. Anything less, and we have contributed to its gradual demise.

We need to focus on the future, by determining how to structure older applications so that they can operate with newer technologies. When we achieve this, we preserve the investment in the COBOL while exposing it to new tools and enabling it to respond more quickly (and with less disruption and less cost) to new requirements in the business.

The Acucorp Vision

At Acucorp we see ourselves as a vendor who enables COBOL sites to move applications forward in exciting ways. In addition to prolonging their usefulness, we offer ways to enhance their capabilities and enable them to interoperate with the newest technologies.

Acucorp customers can deploy their COBOL applications on the Web; interact with .NET applications and services running under Windows; call .NET applications and services from their COBOL programs; and create .NET services for use with their COBOL.

Another significant innovation in our pipeline can help COBOL sites with the mining and publication of the business logic in their applications. When this technology is available, the first step will be for the site to identify and package self-contained COBOL modules, each of which performs a distinct business function such as customer look-up. The focus is on identifying sections of code that solve specific business problems.

These modules (written in COBOL) will become Web services, which can be published, located, and invoked across the Web from any location, allowing the company to transcend hardware and operating system boundaries.

If a site sets out to modularize everything at once, the task may seem overwhelming. But companies don’t have to do it all at once. They can identify Web services as they need them. They can actually accomplish a great deal with just a few services—20 to 30 can drive a Web solution quite nicely—so the total amount of work does not need to be large. Sites can be selective about which COBOL modules they want to expose to the outside world.

Several vendors offer excellent tools that help sort out the structure of an application and document that structure. In addition, software methods and tools that use Unified Modeling Language (UML) may prove helpful for visualizing and documenting the structure of COBOL systems, thus easing the modularization process.

The benefits of Web services apply to COBOL as well as to any other application programming language. It doesn't matter which language the Web services are written in; what matters is that the services use an accepted set of protocols and standards to communicate with the outside world.

Although the move to Web services is a significant step, our tools will help COBOL sites to handle the move in an evolutionary way, at a pace that makes sense for the business. Many benefits accrue from taking this step:

- Applications can make use of their own services. Thus, over time, companies can slowly simplify their own programming model. In other words, the service approach provides a road map for cleaning up application code at the company's own pace.
- The service approach also provides a sensible model for bringing other languages and tools into the programming mix, so businesses can make more effective use of modern programming languages. They can plan a migration path where they can bring Visual Basic, C#, or Java programs to bear on a problem where appropriate.

This is an evolutionary effort instead of a revolutionary rewrite or replacement strategy. By incorporating modern tools through standard approaches, Acucorp's approach will allow companies to turn to these tools when they need them, while still retaining all the hard work they have invested in their COBOL applications over the years.

Programmers writing in Java, C#, or Visual Basic won't even need to know that they are talking to a COBOL service. The two pieces interface through standard protocols, and it really doesn't matter which language is on either end.

- Application resellers can gain new market opportunities by making Web services available as part of the application or by adding e-commerce capabilities that they didn't offer previously.
- Acucorp's "standards-based connector" approach will allow easier solutions to integration projects.

Web services promise to afford enterprises many opportunities. They provide a method for integrating disparate applications, interoperating with different languages and platforms, providing universal connectivity and enhanced data access, and increasing market responsiveness and application ROI.

For enterprises invested in COBOL, I think Web services provide an even bigger opportunity. By slowly transforming legacy applications into COBOL Web services, an organization can open the door for newer technologies to be interspersed with its legacy systems, and thus, companies can breathe new life into proven COBOL code.

Programmers trained in newer languages can develop business services in whichever language they choose, while COBOL developers can continue to develop and fine-tune the logic that runs the business.

As new development requirements arise, companies can immediately call upon existing services to assemble a new application.

At Acucorp, we are committed to extending the breadth and usefulness of legacy COBOL code. We continually strive to provide tools and features developers can use to expose existing COBOL systems in new ways, at a pace that meets the organization's needs. In this way, we can help keep COBOL alive into 2020 and well beyond.