COBOL IN THE INFORMATION SYSTEMS CURRICULUM: A DINOSAUR OR AN ENDANGERED SPECIES?

Dr. Richard Aukerman, Texas A&M University-Kingsville, richard.aukerman@tamuk.edu
Dr. Dennis L. Mott, Oklahoma State University, dmott@okstate.edu

ABSTRACT

The COBOL programming language has been considered by many to be cumbersome and outmoded nearly since its inception. Many replacement languages have been suggested, without success over the years. However, its predicted demise seems to finally be occurring, at least in many schools of business. This paper uses secondary sources to determine if the near-death of COBOL in the information systems curricula is a rejection of its inability to meet current business needs and therefore its diminishing use in industry.

Keywords: computer programming, COBOL, legacy code, Web design

INTRODUCTION

The purpose of any language is to enable communication. Most languages provide multiple methods for communicating the same, or a similar, meaning to the intended audience. The ability to state a meaning using various communication techniques adds flexibility to a language—but it also increases the complexity of that language.

In the case of computer programming languages, a major goal is to create effective communication between people and computers. A programmer's skill in using a specific language provides greater flexibility for solving a problem. This increased flexibility allows the programmer to solve a problem using a method that seems perfectly logical to him, but may be quite unclear to other programmers who will are charged with maintaining that code. This justifies the need for coherent internal and external documentation.

Third generation procedural computer programming languages are still widely used in today's business environment. Perhaps the "granddaddy" of them all is COBOL. This "ancient" language was first standardized by CODASYL (Conference on Data Systems Languages) in 1960, and its demise has been predicted since 1961. Several universities no longer teach the language, partially because it is no longer "in vogue," partially because they have few faculty members that are interested in teaching it, and partially because of little demand from the primary employers of their graduating students. All would probably claim lack of demand as the primary reason for dropping COBOL from their requirements for Information Systems majors, but also would probably not begin offering the language again regardless of employer demand.

The question pondered by many involved in information technology education is whether COBOL should be summarily executed or left to die a lingering death. Some institutions have chosen to immediately eliminate the subject from the curriculum, some are hoping that making COBOL an elective course will lead to low demand from students and thus provide justification for not offering instruction in COBOL. Perhaps we should evaluate the likelihood that COBOL will ever actually offer a practical alternative for enhancing or maintaining productivity before a decision is made about its elimination or continuation.

The Decision Bases
As an English-resembling language, COBOL is highly self-documenting. A well-written program should not need extensive internal documentation. Thus, the coding requirements for maintenance requests can begin earlier than could happen if there was a need for an exhaustive review of internal documentation. The self-documenting attribute of the COBOL language should make it easier to determine which modules in the program contain the code for performing any specific task.

Some have expressed the desire to implement object-oriented systems as an argument for the replacement of COBOL as the programming language of choice. Others would say that we could simply become skilled at object-oriented COBOL, since it does exist and should diminish the learning curve for the move into object orientation by those who already know the language. After all, hasn't COBOL always used some of the concepts of object-oriented programming such as reusable code? Can a traditional module developed in COBOL be considered as an Object? Unfortunately, the answer to the previously questions is probably not an unqualified "yes," but it is enough of a “yes" to cause several software development firms to invest in developing approaches to use COBOL in Web design projects.

Many newly developed applications are designed to be web-centric. COBOL, as a language used primarily for batch-oriented applications, would appear to be out of the loop when determining the best language for these newer applications. It is often considered superfluous, "tolerated only because it is difficult to convert the existing 225 billion lines of COBOL code to Java." (5) Of course, this is no longer a valid argument because tools have become available to simplify the conversion process. The difficulty of a task is reduced when software can perform a percentage of the work that was previously completed entirely by programming personnel.

In 1997, Micro Focus announced the availability of NetExpress. NetExpress "places COBOL in a 'wrapper' that makes it directly accessible to a Web server. This makes the Web suddenly available to the legions of COBOL developers who are not familiar with HTML or Java." (2) As an example, the Army and Air Force Exchange Services in Dallas wanted to give its vendors access to its invoicing system, but its developers were not trained in Java or HTML. Section chief Clyde Todd said his team avoided wrestling with unfamiliar programming languages by using NetExpress to write a COBOL interface that pulls data from a SQL database and publishes invoices in HTML on an extranet accessible to the vendors. (2)

Current economic conditions are resulting in managers rethinking the approach to implementing technological change. Technology for the sake of technology is simply not acceptable. Instead, Return on Investment (ROI) is once again taking center stage as the primary criterion for evaluating information technology expenditures. Re-engineering from one programming language to another seldom results in a short-term positive ROI. Long-term payback for software development costs is, of course, important--but it is more often than not difficult to justify during dire economic conditions. Decisions relative to code maintenance versus code rewrite is arguably difficult to justify; however, the process of verifying that replacing the COBOL code in a currently functioning system with code written in another language is economically feasible is equally difficult to justify. (6)

Other software is also being developed to address the issue of COBOL versus "something else" and making it become COBOL and "something else." For example, Cape Clear Web Services Software has developed software to convert COBOL applications into a Web Service without modifying the code.

LegacyJ Corporation currently markets software that uses COBOL as the web-enabling syntax. This has been done while maintaining consistency with the new ANSI standards. This software is touted as eliminating any requirement for COBOL programmers to learn Java or CGI in order to make use of the capabilities that these technologies make available. (4)

Management priorities for spending on technology have also changed in relation to changes in the external environment. Much greater emphasis is now being placed on backup of data, programs, and even hardware. Given that a finite quantity of funds are available for use in information technology, the benefit to be realized from replacing existing code with a coding language that is more "current" is not only difficult to economically justify but is often considered less critical for the attainment of long-term strategic goals and objectives than are other technology expenditures.

A discussion of the practical applications of Java and COBOL should make it clear that the two languages should not be evaluated as competitors. A comparison of the two is similar to comparing a defensive football unit with an offensive football unit. They are both attempting to help the team win the game and they are both necessary, but they perform completely different functions and (despite the same ultimate goal) serve completely different purposes. Tom Yancey, General Manager of Fujitsu Software, has stated that "...no one single programming language will ever provide a universal solution for creating all software. History has proven this time and time again. In the 1970s, PL/I was said to be the universal solution. In the early 1980s it was BASIC, then in the late 1980s it was C. In the early 1990s it was C++, and now in the late 1990s, Java zealots believe it to be the universal solution for creating all software." (7)

In an interview with a software engineer from IBM, Edward Hurley, Assistant News Editor for Search390 queried about when COBOL programmers should learn Java. The response: "Only when they need to. COBOL may be around long after Java because it has such a huge installed code base. Sure, Java is hot, sexy and gets all the press, but we heard for years that COBOL would be replaced. First it was PL/l, then Pascal, Smalltalk and C++. You will probably hear that COBOL will be replaced by what follows Java as well." (3)

Martin Pagnan, a freelance software architect and application developer from Quebec, observed the following, "When it comes down to the question of which will best survive, the bottom line is that COBOL can be much more easily enhanced to include web features than Java can be enhanced to handle records. Adding web routines to COBOL is being done; adding record handling to Java is not and cannot be done, without a major change to Java's restrictive object orientation philosophy." (5)
While it seems that some are striking up the funeral dirge before the death has occurred, others may have an exaggerated concept of COBOL's usefulness. Burger stated that "COBOL programmers will be the single greatest influence on the future of Linux and Open Source in

general …. COBOL programmers will use the new COBOL tools now available to take Linux and Open Source into the twenty first century, making the world's most popular business programming language the technical leader as well." (1) The capabilities of COBOL will have to evolve significantly before large numbers of information systems professionals will agree with Burger's comments.

SUMMARY

"COBOL has been given a bum rap as being old and out of date. It is true that COBOL has been running our world's businesses for almost 40 years, but that may be its strength and not its weakness. COBOL was created to accomplish the uninteresting, but very important task of bringing life to the business processes. COBOL was designed to be verbose and code to be self-documenting. Data is defined within the COBOL program and the program can be moved among a wide variety of operating platforms. The rap is not that COBOL doesn't do its job; it is that graphical applications are now in vogue." (4)

Ultimately, we should be asking ourselves not what is the best language, but what is the best language for the given problem scenario. Professionals in any field try to select the best tool available for the task currently being performed. If a new tool is better than a tool previously used, the professional will make the change. However, they do not simply buy new tools because new tools are available. As educators, we owe it to our students to prepare them for the widest variety of potential coding situations as possible. Aspiring computer professionals need to know how to use the newest tools available, but not at the expense of knowing how to use a variety of tools. Most indications are that COBOL should continue to be one of those tools that should be in the mix of those presented to our students.

REFERENCES

1. Burger, T. W. "COBOL in an Open Source Future." January 7, 2002. Retrieved August 27, 2002 from http://objectz.com/columnists/tw/01072002.asp
2. Dalton, G. "COBOL to the Web." Information Week OnLine, September 22, 1997. Retrieved March 3, 2003 from http://www.inforrnationweek.com/649/49iucol.htm
3. Hurley, E. "COBOL and Java: A Programmer's one-two punch." May 22, 2001. Retrieved February 27, 2003 from http://search390.techtarget.com/qna/0,289202,sidl0-gci553985,00.html
4. LegacyJ Corporation. (2000) "Web Enabling COBOL." Retrieved March 3, 2003 from www.legac3d.com/COBOL/COBOL-web.html
5. Pagnan, M. (2002). "Can A Java Programmer Be Transitioned to COBOL?" Retrieved August 27, 2002 from http://objectz.com/columnists/martin/02252002.asp
6. Turner, S., Aukerman, R, & Walstrom, K. "Breakeven Analysis Applied to Software Maintenance Decisions: A Theoretical Paradigm." Southwest Business Review, 3(1), 1993, pp. 113-126
7. Yancey, T. "COBOL is the Enterprise." Retrieved August 27, 2002 from http://objectz.coni/COBOLreporUarchives/vieW_yancey.htm