The Resurgence of COBOL: Considerations for the Future

ABSTRACT: The COBOL language is alive and well. There are those who deny this and contend that 'COBOL is Dead.' There are five important factors which contradict that statement. First, industry is finding a resurgence in the importance and use of COBOL. Second, COBOL's environment has switched from mainframe to the distributed level of PCs. Third, academic institutions have also changed to teaching COBOL on the PC level. Fourth, additions and refinements to the COBOL syntax show an evolutionary development is in process. Last, continued improvement in the standards and cooperation between CODASYL, ANSI and ISO help creates a language that is accepted not only in the United States but also internationally. These factors all contribute to the continued use, education and enhancement of the COBOL language.

Dr. Ronald J. Kizior
Management Science Department
Loyola University Chicago
Chicago, Il 60611-2103
1-312-915-7394
#w40rk0@luccpua.it.luc.edu

KEYWORDS: *COBOL, Curriculum, Object Orientation*

INTRODUCTION

COBOL has had more impact on business than any other programming language. The sheer amount of code that has been developed over the past 34 years is a fitting testimony to the acceptance of this language. However, there are some programming professionals who say that COBOL, regardless of its successful track record, is waning and "Dead." This paper will show that this statement is the farthest thing from the truth. COBOL, since its inception, has undergone various revisions, attesting to the fact that it is alive and willing to comply to user demands. COBOL is widely accepted within the continental U.S. as well as internationally, as documented by the existence of numerous user groups.[1] [1]

WHERE IT ALL STARTED

COBOL had its official and unceremonious start on April 8, 1959. This date is usually regarded as the birthday of COBOL. This date was chosen because it signifies the day that a meeting convened at the University of Pennsylvania by the Department of Defense (DOD) to define the objectives for a new Common Business Language. Captain Grace Hopper, the mother of COBOL, was the person in charge of this group. [2]

This language originally had no formal name, and it was left up to the group developing the language to formulate an appropriate moniker. Some attempts were BUSY, BUSYL, INFORSYL, COSYL, DATASYL and COCOSYL. When the final draft of the language was completed in December 1959, COBOL emerged as the consensus name agreed upon by this group. The exact deliberation as to why this name was chosen over other alternatives is not known.[3]

COBOL has gone through various versions (COBOL-60, COBOL 68, COBOL-74, COBOL-85), plus other minor revisions, and has been subjected to rigorous standards imposed by ANSI (ANSI/X3J4) and ISO committees (ISO / SC22 / TC97 / WG4). COBOL-85 has evolved from the work of three major committees: CODASYL (Conference on Data Systems Language) COBOL committee, ANSI/X3J4 committee and ISO (International Standards Organization) [4]. As the following diagram shows, these committees have an integrated relationship.

CODASYL <———> ANSI <———> ISO

Once the CODASYL committee develops new standards, they are then submitted to the ANSI/W3J4 COBOL committee for additional review. When agreement is reached by the ANSI committee, they, in turn, pass them on to the ISO(WG4) COBOL Committee for approval on an international scale. As each new version of COBOL is approved, new "functionality" is added in order to increase the language's flexibility and acceptability.[5] This is demonstrated by the material presented in the next several paragraphs.

EVOLUTIONARY PROGRESS

COBOL has flourished over the years by incorporating the structured methodology that came out of a special CODASYL committee symposium in the early 1970's. Such features as Scope terminators, nested programs, EVALUATE verb, in-line PERFORM, DO-WHILE and DO-UNTIL are the result of that symposium.

The most current version, COBOL-85, emerged after a long delay, which many felt was damaging to the language. The CODASYL COBOL Committee held a syposium in Los Angeles in 1975 soliciting suggestions from all over the world on how to change COBOL in order to accommodate structured programming techniques. Most

1. Micro-Focus has user groups established in Amsterdam, Australia, Canada, Finland, Italy, Norway, Spain, Germany and United Kingdom.

of the new features incorporated in COBOL-85 had their roots at this symposium. [6] In order to ensure that this delay would not occur again , the International COBOL Committee met in Vienna, Austria, in 1984 and adopted new rules, namely, an Addendum Process. This process provides the mechanism by which new features can be added to the COBOL language without causing incompatibility with older COBOL programs. In September 1989 the committee passed an Intrinsic Functions Addendum , which resulted in COBOL-89 being issued. This addendum added forty-two new intrinsic functions to the COBOL language. A few examples of the functions that were added are ANNUITY, CHAR, CURRENT-DATE, LOWER-(UPPER-) CASE, MAX , MIN , REM , REVERSE, SUM, SQRT , various calendar functions and other popular mathematical and trigonometric functions. [7]

Some of these built-in functions operate in the manner shown in the chart below. (Lower case identifiers are user defined.)

Assuming that the three test scores are 82 , 79 and 93 respectively, the minimum score value is 79, which is not less than 70, and the phrase 'STUDENT HAS PASSED ALL EXAMS' will be displayed. Notice in the chart, that in order to activate a built-in function, the keyword FUNCTION must precede the unique function name.

```
COMPUTE Month-Payment
= Prin-Amt * FUNCTION ANNUITY
(Monthly-Int-Rate, No-months)

END-COMPUTE
```

Assume the following data for these variables:

Prin-Amt =	Principle amount of money invested $100,000.
Monthly-Int-Rate =	Monthly interest rate, i.e. if annual rate were 8.85, then the monthly rate would be .0074.
No-months =	Number of months for this annuity, i.e.15 years times 12 months equals 180.

The actual syntax using the previous data would be

```
COMPUTE
Month-Payment= 100000 * FUNCTION
               ANNUITY (.0074, 180)

END-COMPUTE
```

Another one of these built-in functions is the MIN function, and it works like this.

```
IF FUNCTION MIN (TEST1,TEST2,TEST3)
NOT LESS THAN 70 DISPLAY " STUDENT
HAS PASSED ALL EXAMS".
```

The ANSI COBOL committee went to work right after the COBOL-85 version was released, and is currently working on the next addendum, which, when released could result in COBOL-95/96. The committee is currently looking into the possbility of adding additional flexibility to the language by introducing such syntax as EXIT-PERFORM, extensions to the ACCEPT and DISPLAY verbs, windows support, multi-tasking, boolean/bit functions, extensions to the CALL statement and other features that will allow object oriented programming.[8]

WHAT DOES THE FUTURE HOLD

The future should bring increased interest in COBOL because of the possibility of incorporating OOP (Objected Orientated Programming) techniques into the language.[9] The CODASYL committee, in Scottsdale, Arizona, in November 1989, formed the OOCTG (Object Orientated COBOL Task Group) (ANSI/X3J4.1) to oversee this project. [10]

Another interesting change that has recently occurred is in the COBOL platform. COBOL was originally used in a mainframe environment and was always thought of in that light. Over the past three years, however, there has been a steady change down to the PC level. Because of improved technology and software availability, more companies are now considering PC-based COBOL.

CORPORATE CHANGE

Several years ago Frito-Lay, based in Plano, Texas, had a distributed platform that linked GUI-based PCs to their massive corporate database which resided on a mainframe. The original distributed systems platform was primarily written in 'C.' This architecture was not interactive and ,therefore, not fully satisfying the users' needs.[11] Since they had experience with host COBOL, Frito-Lay decided to use this experience to build a PC-based COBOL development environment that would be the basis for their new distributed computing system. By choosing COBOL, Frito-Lay was able to avoid encountering a big learning curve .

Frito-Lay's distributed processing system was actually made possible by using an object-oriented programming approach.

There are various classes and methods of data. These data are passed back and forth among the distributed platforms. The sales and marketing departments were the only departments that used the original distributed system. Now other systems such as personnel, payroll and manufacturing have joined the request bandwagon.

The improved efficiency measured in terms of 8 to 10 months request time under the old system has now been reduced to a matter of weeks under this PC-based distributed system. The new applications also provide end users with the added advantage of working with a GUI interface. [12]

ADDITIONAL INDUSTRY CASE STUDY

Any person remotely familiar with the activities of an insurance company realizes that there is a huge amount of information processing that goes on daily. Insurance companies, and other companies like them, face a continuous problem of program maintenance. Production programs in the insurance industry are constantly being affected by various changes in federal and state laws, and by various insurance regulatory agencies. In order to stay competitive in today's markets, these companies are always looking for new flexible computing alternatives. The Leverage Group of Glastonbury, CT, is one such company that has been using PC-based COBOL to maintain a competitive edge and implement new technology while preserving investment in existing applications and skills.

The Leverage Group was originally a mainframe shop, but, in order to create their own market niche, they wanted to develop a PC based solution that would allow insurance companies to automate policy administration on a PC network while continuing their mainframe development. One of their core products contains 350,000 lines of code and is targeted to run on DOS, MS-Windows or OS/2 operating systems and on a Novell LANserver or LANmanager network. Those companies who have used The Leverage Group's ALS (Administration Leverage System) product have found this product to be both cost effective and flexible. Working in a PC environment allows The Leverage Group to compete with mainframe vendors because modifications can be done faster and cheaper. [13].

4GLs CHALLENGE

The previous two corporate case studies help point out two important facts: 1) PC-level implementation, and 2) the use of COBOL for that implementation. The proponents of 4GLs have long praised the speed and productivity increases of their products. Their claim is how can anyone today use a 3rd generation language competitively? They voice the opinion that "COBOL is DEAD."

For the non-technical person, one of the advantages that 4GLs have is their speed of development. In that process there is a drawback. The drawback is that 4GLs are interpretive rather than compiled. This reduces their speed advantage when compared to COBOL. Also, as stated previously, 4GLs , unlike COBOL, lack standards.

Trying to change all of the existing COBOL code, estimated today at over 70 billion lines, [14] into a 4th generation language is definitely a challenge, if not an impossibility. What is needed is some kind of cost effective conversion technology, which to date is non existent. This, by far, is the major stumbling block for most 4GL vendors. This particular fact provides an excellent reason why COBOL is neither dead nor on the verge of dying. [15] There is a definite need for programmers to maintain this library of production programs written in COBOL. Industry would come to a gridlock situation if there were no COBOL programmers available to maintain the existing library of source code.

Ms. Kimberely Saxine, the manager of development consulting for Hewitt & Associates, believes that COBOL is far from dead. She says,' I think you're going to find a lot of uses for it (COBOL) just because there's such a large base of mainframe COBOL programmers out there. You just can't throw it away."[16] She further contends that the high development cost and reputed unruliness of your typical 'C' program are not even being considered.

John Crabtree, a programmer/analyst for Tandy Corporation in Forth Worth, Texas, concurs that COBOL is a standard in their shop. He contends that you can find experienced COBOL programmers nationwide, which is not the case for 'C' or Smalltalk developers. [17]

THE FUTURE IS 'COOL'

'COOL', Cobol Object Orientated Language, actually exists today on a proprietary level. Hewitt & Associates, a large information systems consulting company in Lincolnshire, IL, has developed its own version of OO-COBOL. They use 'COOL' to build reusable modules of code for productivity and efficiency. Their developers run 'COOL' through a PC-based precompiler that generates OO source code that is then submitted to a regular PC-based COBOL compiler. [18]

Standardized 'COOL' is in process. This process started at the symposium sponsored by the CODASYL COBOL Committee in November 1989 in Scottsdale, Arizona. The purpose of this symposium was to explore the relevance of object-oriented methodologies to COBOL. The OOCTG was formed at this meeting with the eventual goal of creating standards for OO-COBOL. Their agenda, as originally planned, is shown in the chart below.

OOCTG AGENDA	
OOCTG new OO final syntax	March 93
Early edition of OO-COBOL	Fall 93
Official OO-COBOL Standard	Somewhere 1994-95
Next Full revision	1996-97

Some of areas in which major discussion has to take place with regard to 'COOL' are Orthogonality, Recursion, Polymorphism and Encapsulation. In an attempt to derive 'COOL' standards, the following goals were created: [19]

- Compatibility - with existing COBOL syntax
- Simplicity - include obvious semantics with simple syntax
- Consistency - well-defined COBOL syntax will not change
- Maximum language power - more operations per keyword
- Readability - always an objective since 1959
- High Performance - in run, compilation and development time
- Transformability - ability of an object to adapt to different uses in different environments with minimum effort.

The first date on the above agenda has been met and completed on time. The second item has been developed, and formal approval is expected shortly. A new book by Raymond Obin has recently been published by Micro Focus Publishing entitled *Object Orientation: An Introduction for COBOL Programmers*. This text explains various teminology that can be used in conjunction with 'COOL', but, again, at this time it is all unofficial and subject to further modifications by the standards committee.[20]

NEW BREED OF COBOL COMPILERS

Drake Coker, the chief scientist and author of Acucobol-85, points out that the new versions of COBOL are slowly acquiring various development tools, such as GUI and translation codes for SQL. Screen design, historically difficult using COBOL, is also much quicker and easier with the latest version. The field is expanding. Currently, the big names in COBOL compilers are RM/COBOL, Acucobol-85, Micro Focus COBOL, CA-Realia COBOL, Visual-COBOL and Microsoft COBOL V.4 .[21]. A brief warning to those in the market for a new COBOL compiler: Check out each vendor's product to determine whether it is a true compiler or an interpreter. For example, Micro Focus Cobol is a true compiler; Acucobol-85 is an interpreter. [22,23] The biggest difference between interpreters and compilers is in terms of efficiency in execution time, of which the later is superior. Interpreters translate one instruction of source code into machine language and then immediately execute that instruction before going on to the next instruction. Compilers translate the entire set of source code instructions into machine code, which is stored in a separate file, and then execute the entire set at one time.[24]

The previously mentioned vendors are making their products available on various platforms, such as DOS, OS/2, VAX/VMS and UNIX. They are also available across various GUI-based systems, such as WINDOWS, OS/2 and OSF/Motif. This flexibility has cut deeply into the claims of the various 4GL power tool developers. [25]

Another point in favor of COBOL is the fact that it is a vendor-independent standard. 4GLs have failed to dominate the development market because no one language has emerged as an accepted standard. Even though the competition admits that its new 4GL products are used more for downsizing, a major portion, 80% as claimed by Drake Coker, chief scientist at Acucobol, Inc., of all applications is still written in COBOL. [26]. Rich Wunder of CSC Partners, an information systems consulting company in Newton Lower Falls, MA, claims that as many as 65% of all corporate systems are still running in COBOL.[27] Whether 80% or 65%, there is still a healthy portion of corporate systems using COBOL.

This demonstrates the increase in productivity that can occur in an academic setting by switching to a PC-based COBOL environment from a mainframe environment.

ACADEMIC RESPONSE

Given the fact that many industry environments have changed from a mainframe to a PC-based standard, the academic community has had to take a long, hard look at what they are going to do in terms of teaching COBOL. Various book publishers have helped ease the transition by having available, as supplementary material, student versions of PC-based COBOL. Other vendors have created grant programs so universities can obtain a regular version of their PC-based COBOL compiler at a reduced rate. Still others have developed student versions of COBOL compilers that sell for $49.95. These student versions usually have a limitation in terms of the size of the file which they can handle.

Some schools have been working with PC-based COBOL for two to three years already, and have been very successful. One community college in the Chicagoland area switched from their mainframe COBOL to PC-based COBOL in September 1992. In the past, 8 assignments were a required part of the course. When the second semester started in January 1993, after having experienced one semester, 10 assignments were made mandatory, and in September 1993 the instructor raised that to 12 assignments. This demonstrates the increase in productivity that can occur in an academic setting by switching to a PC-based COBOL environment from a mainframe environment.

CONCLUSION

It is obvious from what has transpired recently that COBOL is not dead. The single most popular commercial programming language today is COBOL, which accounts for somewhere between 65% and 80% of all commercial applications.[28] As was pointed out previously, there seems to be a renewed interest by numerous firms to use their COBOL experience in tackling new endeavors. With the introduction of the newer and more powerful 486 PCs, companies have taken a harder look at distributed platforms while continuing to use their mainframe COBOL experience. Although COBOL may not be popular on numerous college campuses now, the best of what COBOL has done and what it will do in the future is yet to be made known. Increased interest in PCs combined with the capability of learning COBOL on the PC level should provided the stimulus for increased interest in the language. It is, therefore, our job, the job of the college professor, to take this information and make it known to our students. We are being called upon to enlighten and prepare the youth of today for the careers of tomorrow.

REFERENCES

1. *Compilations*, May/June 1993, p.12.
2. Garfunkle, Jerome, "COBOL in the 90's". Seminar presentation, Ramada Hotel, Rosemont, Il. Sept. 9, 1992.
3. *Ibid.*
4. Schreider, Don ,"The Organizations behind the Acronyms", *Compilations*, March/April 1993 , p.10.
5. Garfunkel, Jerome, "IN DEPTH - COBOL - The Next Stage," *Computerworld*, July 23, 1990, p.88
6. *Ibid.*, pp.87-8.
7. Garfunkel, Jerome, "COBOL Rescues The Millennium", *Enterprise Systems Journal*, October 1991, pp.36-7.
8. *Compilations*, Nov/Dec 1993, p.19.
9. Urlocker, Zack, "The Future of Object Orientated", *Chicago Computing*, April 1989, p. 13 .
10. Schricker, *Op. cit.* p.10.
11. Kalkowski, Rick, "Frito-Lay Uses Micro-Focus COBOL to Make Corporate Data More Accessible", pp.9 , *Compilations*, March/April 1993.
12. *Ibid.*, p. 10.
13. Kalkowski, Rick, "The Leverage Group Offers Insurance Industry an Efficient Mainframe Alternative", p.7, *Compilations*, January/February 1993.
14. Jenkins, Cameron, Vice-President, Marketing, Letter dated January 22, 1993, Acucobol, Inc.
15. Sokol, Marc, "Why 4GLs Cannot Kill COBOL", *Journal of System Management*, May 1990, p. 35.
16. Snell, Ted, "Are you Ready for Cutting Edge COBOL?", *Datamation*, Oct. 15, 1992, p. 77.
17. *Ibid.*, p. 77.
18. *Ibid.*, p.77.
19. Garfunkel, Jerome, *Op. cit.*, July 23, 1990, p.87-8.
20. Obin, Raymond, Object Orientation: *An Introduction for COBOL Programmers*, Micro Focus Publishing, 1993.
21. "Who's Who is PC Cobol", *Datamation*, May 15, 1991, p. 72 .
22. Celko, Joe, "Queries and COBOL", *Database Programming & Design*, March 1992, pp.26-7.
23. "Off-Mainframe Compilers Do the Job," *Corporate Computing*, Dec. 1992, p.97 .
24. Ross, Steven C., *Understanding Information Systems*, West Publishing Company, 1994, p. 11.
25. Snell, *Op. Cit.*, p.78 .
26. *Ibid.*, p. 82.
27. Colborn, Kate, "Cobol and Beyond", *Datamation*, January 15,1992, p.65.
28. *Ibid.*, p.65 .

Author's Biography

Ronald J Kizior, assistant professor, of information systems at Loyola University Chicago. He received his B.B.A. in Accounting from the University of Notre Dame, M.B.A. in Transportation from Northwestern University, and his M.A. and Ph.d. in Economics from the University of Notre Dame. He has taught at Loyola since 1969. He is active in EDSIG, the Special Interest Group for Education which is a part of the DPMA. He has held all of the elected positions in that organization, and is currently the Director of Membership Services. He has chaired the ISECON conference in 1988 and 1990. He has had presentations at the Midwest Business Administration Association Conference , the Midwest Computer Conference and at ISECON. His current areas of research and interest are in COBOL, Quality Systems Development and International Information Systems.

ISSN 1055-3096