https://doi.org/10.48009/4_iis_2018_87-95

Issues in Information Systems

Volume 19, Issue 4, pp. 87-95, 2018

STILL TEACHING COBOL PROGRAMMING – UNDERLYING REASONS AND CONTRIBUTING FACTORS

Azad Ali, Indiana University of Pennsylvania, azad.ali@iup.edu
David Smith, Indiana University of Pennsylvania, david.smith@iup.edu
Andrea Morman, Indiana University of Pennsylvania, a.d.morman@iup.edu

ABSTRACT

This paper discusses and analyzes the reasons for continuing to teach the COBOL programming language despite the many opposing factors. The paper focuses on the case of a particular department in Computer Science at a university located in Western Pennsylvania that has been teaching COBOL for years and intends to teach it for the foreseeable future. The paper reviews literature and debates in favor for and against teaching COBOL in today's curriculum. It will then shift the focus on the market and discusses the reasons that the market continues to use COBOL despite the mounting pressure to replace it. The case of the department of computer science and their justification for continuing to teach COBOL is presented last.

Keywords: COBOL Teaching, Legacy Teaching, Programming Language, Mainframe Teaching

INTRODUCTION

> The crisis in college and university COBOL education is becoming more acute as time passes. The reduction and total elimination of COBOL courses, declining interest by both faculty and students, and low visibility of new commercial applications being developed using COBOL are commonplace. Yet many practitioners are well aware of the billions of lines of existing, operational, revenue-producing lines of COBOL code that continue to provide the mainstay of many commercial enterprises. (Roggio, Comer & Brauda, 2003, p. 3).

Although the above quote was written more than fifteen years ago the same argument surrounding the discussion of teaching COBOL still exists. The landscape of programming languages is in a state of constant change and the number of new programming languages and related technologies are on the rise (Ali & Smith, 2014). At the same time, some older programming languages (termed as legacy) have slowly diminished in the industry and vanished from academic courses. Khadka et al. (2014) listed some of these legacy languages (e.g. RPG, VB, and PLI) that were once popular but now they have all but disappeared from computing curriculum. Other programming languages (like Ada, Pascal, Fortran and others) are hardly mentioned in terms of uses at academic and professional settings (Parker, Ottaway & Cho, 2006). Yet COBOL has been resilient within industry for decades and it remains in the curriculum of some academic institutions as a course within a program.

One of the universities that keeps teaching COBOL is located Western Pennsylvania. The department of Computer Science (CS) at this university has been teaching COBOL in one of their programming courses for over forty years and intends to continue teaching it in the foreseeable future. The department has their own reasons and justifications that contribute to their decision for teaching COBOL and these reasons are explained later in this study. However, some background information and review of literature is necessary in order to fully understand this case, thus this paper intends to present the current role of COBOL in industry and reasons for and against teaching it in a computer science program.

The remainder of this paper is divided into the following:

- The paper begins with an explanation of the reasons that stand against the teaching of COBOL.
- The second section talks about the opposite side of the coin. It explains about the factors that stand in favor of teaching COBOL.

- The next section gives discusses the reasons that the industry keeps using COBOL in their applications. This is critical because it is the primary reason for continuing to teach COBOL at academic institutions.
- Lastly, the paper explains the case of this paper from the perspective of a department that is still teaching COBOL together with their explanation and justifications.

THE MOUNTING FACTORS AGAINST TEACHING COBOL

There are different factors that stand against teaching COBOL and these factors are increasing. At the same time, pressure is mounting on academic institutions to drop COBOL from the curriculum in order to meet accreditation requirements while retaining more modern programming languages (Roggio, Comer & Brauda, 2003). Along the same lines Ali and Smith (2014) listed some of the reasons that stand against using COBOL, including the dislike of the language by the students, the seemingly dwindling market base for jobs that require COBOL experience and the perceived dismal future for the COBOL language. Sagers et al. (2013) also noted that faculty interest in teaching COBOL is dwindling. Due to many factors that stand against teaching COBOL, further explanation and discussion is warranted. The remainder of this section elaborates on some of these factors.

The Dislike for COBOL

Roggio, Comer and Brauda (2003) noted that COBOL can get as much 'uncool' as possible among the students. That is the dislike of the student for COBOL was big and continues to increase. KVSN and Choudhari (2013) used an analogy when comparing COBOL to other modern programming languages by asking the question "can the elephant run with the deer?" Dunn and Legerfelt (2005) suggested that COBOL is too wordy and out-of-date. Paulson (2001) expected that COBOL "is going out the way of Latin, obsolete save for some critical text" (p. 13).
Phillips et al. (2013) explained that most students are not motivated to learn COBOL. The same authors attempted to motivate the students to learn COBOL and acquire the mainframe skills. The authors enrolled their students in the IBM Master of the Mainframe Contest (IBM, 2017). Yet despite their efforts, they noted that out of the over 200 colleges and universities in Texas, only 8 participated in the IBM academic initiative to teach mainframe skills. This short list of colleges (8 out of 200) indicates that motivating students to learn COBOL is a hard sale.

Parker and Ottaway (2006) used the term "academic acceptance" to describe the word of "likability" for students and noted that students most likely would rather learn newer languages as they emerge instead of keeping the same language.

The likability (or lack of it) of COBOL makes it hard to teach. Programming languages are in general are not easy to digest among majority of students (Westfall, 2001). Added to this the wordiness of COBOL and the type of applications implemented in COBOL make it very unlikable among students (Ali & Smith, 2014).

The applications that are typically processed in COBOL (batch, reports, file processing and others similar applications) do not fit the applications that are typically suggested to grab students interest in programming. Leutenegger and Eddington (2007) suggested introducing game approach to promote programming courses among students. But COBOL is not conducive to game programming given its emphasis on business processing. So, the dislike of COBOL is hard to overcome.

The Dwindling Market Base

Practitioners in the IT field classify application development into two general categories: developing new applications (new development) and maintaining old programs (maintenance). While there is an abundance of maintenance on old COBOL programs, few new applications have been developed with COBOL and limited new development with COBOL is expected in the future (Sagers, et al., 2013). Thus, it is expected that future COBOL programmers will continue on the maintenance of the current set of applications.
So, the analogy of this kind of thinking goes like this: No new development in COBOL and only maintaining old programs. Which means that maintaining old programs will keep dwindling down which means also jobs for COBOL will slowly disappear in the near future. This is presented in figure 1.

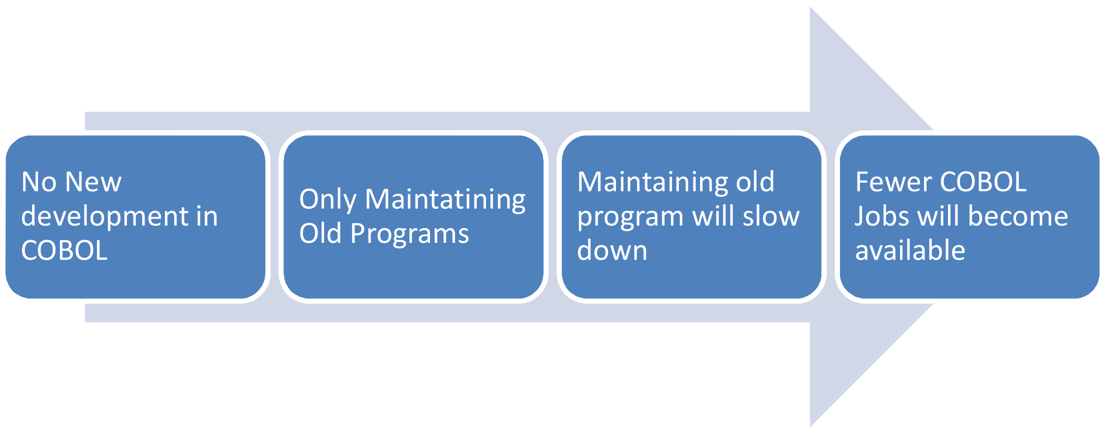
No New development in COBOL
Only Maintatining Old Programs
Maintaining old program will slow down
Fewer COBOL Jobs will become available

Figure 1. Perceived Future of COBOL Jobs

Although there is still demand for COBOL, the demand is limited to mainframe jobs. Even with these mainframe jobs, some companies are moving away from COBOL in favor of other programming languages (Sagers, 2013). In other words, the market base for COBOL is limited to one area: the mainframe. This is contrast to modern programming languages such as Java, Python, and C++ where the market base ranges from supercomputer to yes mainframes but on to web server clusters, desktop computing, to mobile computing, and to embedded computing including the emerging Internet of Things (IoT).

Dwindling Faculty Interest

Roggio et al. (2003) noted that the issues surrounding COBOL programming have resulted in decreased faculty interest in teaching or using COBOL in their classrooms. Likewise, Sager (2013) noted that faculty interest in teaching COBOL is dwindling. Faculty interest in teaching any topic is important but faculty interest in teaching programming language takes another dimension. Programming languages are known to be difficult to learn by students (Ali & Shubra, 2010). Furthermore, the task of programing is known among students to be dry and lacks social interaction thereby impeding a student's motivation to learn. To counteract this issue, faculty turn to new approaches such as "Game First" as a means to motivate students (Leutenegger & Edgington, 2007). However, COBOL is not conducive to these new approaches thereby limiting faculty interest.

Although the responsibilities of faculty in higher education members include research, service and teaching, teaching is always perceived as the first duty. This view is shared by the students more than others. Wardlow and Johnson (1999) noted on another survey that showed 96% of students consider teaching as the prime function of college faculty. Yet if taken with dwindling interest, this prime task of the faculty with lower interest ought to influence the quality of course being taught. Given difficult content to teach, bad publicity and dwindling interest of faculty in teaching makes COBOL that much more unpopular among students.

The Setbacks on Modernizing COBOL

Programming languages in general have gone through different phases of modernization. Program modernization in general went through the following stages:

- Started as top-down procedural programming (Blansit, 2012)
- Evolved into structured approach to coding which helped with the organization of the overall program (Van Gelder, 1977)
- Object Oriented Programming (OOP) was introduced where the focus is on creating objects (nouns) that combine behavior and data as a unit (Blansit, 2012). The unit can then be manipulated and be reused within a program and in other programs (Ehlert & Schulte, 2010)
- Then the point-and-click and GUI programming approach was introduced and interactive programming approaches become popular (Bishop & Horspoo, 2004)
- The web was introduced and the different interfaces that resulted from which provide direct communication with the user (Smith & Ali, 2014).

Although some programing languages evolving over time by adding modern features, COBOL was not able to keep up with the advances (Mitchell & Keefe, 2012). For the most part COBOL remained at the structured approach. Efforts that were put forward to move to the next levels did not materialize. OOP in COBOL was introduced but was not embraced by the industry. Similarly, interface with the web, the role of COBOL program remained at the back end where the front end of the work is typically handled by other programming languages (Gholamin, et al., 2017). At the same time, the point and click approach of other programming languages is hard to integrate into COBOL due to the type of applications that it handles (like batch, reporting, and file processing).

REASONS FOR KEEP TEACHING COBOL

Ali and Smith (2014) listed three factors that stand for keep teaching COBOL: market demand, suitability for the job and association with the mainframe. Dunn and Legerfelt (2005) went a step further and suggested that COBOL cannot be written off as a dead language because the market depends too much on COBOL. Kizior, Carr, and Helpren (2005) indicated that the vast volume of work that is still being done in COBOL makes it impossible to write it off in the foreseeable future.
Aside from these points, there are specific characteristics in COBOL that make it unique in their suitability for certain applications and thus for teaching in certain courses (Mitchell, 2006). The remainder of this sections explains further the factors that stand in favor of keep teaching COBOL in programming courses.

Market Demand

Market demand for graduates with experience in certain programming language is important in selecting a language. Parker and Ottaway (2006) consider this market demand as one of the prime factors for selecting a programming language over others. This is true in all other programming languages. If there is a strong market demand for skills in certain language, this strengthens the likelihood that the programming language will be taught at academic institutions. Given the good prospects of employment on graduation makes it more likely that students will be interested in taking such a course (Machuca & Solarte, 2016).
In the case of COBOL, there are two dimensions can be added to the market demand. There is no doubt that there is a strong demand for graduates with knowledge of COBOL because of the billions of lines of COBOL code that need to be maintained (Barnett, 2005, Kizior, Carr & Halpren, 2005). The first added dimension here is that most of the existing workforce in COBOL is aging and is likely to retire in the next years (Hajnal & Forgacs, 2012). Second, college programs that used to teach COBOL are no longer teaching COBOL given the expanding technologies involving new programming languages and paradigms all within accreditation constraints. In all, there is a shortage of graduates who know COBOL (Mitchell and Keefe, 2012). The market needs more graduates trained in COBOL than is being produced by colleges and universities.

Reasons Initially Cited for Teaching COBOL

At the time of initial development of programming languages, when there were only a in the market, advocates used to cite two reasons for selecting COBOL for their programming courses: first the English like words that describes COBOL statements and sections, and second, the structure of the COBOL language in terms of divisions and hierarchy make it more understandable for some. At the time, COBOL was uniquely placed and have no competition from other languages for these two characteristics. The interesting fact is that these two characteristics still places COBOL uniquely in the market. In other words, despite the many programming languages that were developed since the early days of programming, COBOL is still the programing language best leverages the English Language while at the same time provides the organization and structure for program development.
The English like statements makes COBOL more understandable than others (Stern & Stern, 2002). Although this may sound trivial, no other notable language has similar use of an English like syntax and meaning. In other words, despite the numerous programming languages that were developed, there is no programming language that uses long statements similar to COBOL. So, for students who like English like statements, this may be a good reason for keeping COBOL.

To the contrary, more modern languages are moving away from long statements to brief statements that extensively use special characters, symbols, and notations (like !=, ++, ->, brackets, curly brackets, …) to infer meaning. Statements like "Move A to B", "Add Quantity to Total", and "Set Index up by 1" are abbreviated "B=A", "Total += Quantity", and "Index++" respectively. Furthermore, placement of notations in modern languages have different meanings. For example "++" before or after a variable. These special characters and notations contribute to confusion for a lot of students especially at time when they are still struggling with the core concepts of programming. By using COBOL, a student's fundamental understanding of programming can take root before being faced with plethora of confusing special characters, symbols, and notations.

The Structure of COBOL Programs

Structure of COBOL makes it uniquely qualified to be understood by people who like structure. The four divisions in COBOL programs along with the different attributes that belong to each makes it more understandable than others (Stern & Stern, 2002).
Structure often has a good connotation. A structured approach to development means systematic, better planned and more known steps to follow. In this context, COBOL is perhaps the best programming language that uses the structured approach in which to write programs. The interesting point is that there is no newer language with a similar structured to that found in COBOL.

REASONS INDUSTRY KEEP USING COBOL IN THE INDUSTRY

Academic institutions are not the only side that is debating the issue of whether to continue with COBOL or replace it with another language. The industry is grappling with this issue as well and yet in many cases the industry continues to use COBOL. This section explains some of the reasons that lead the industry to continue to use COBOL.

The Enormous Volume of Maintaining Existing Applications

Fanelli et al. (2016) estimated that there are estimated 180-200 billion lines of code are still in use today as companies attempt to modernize their old systems and move to the cloud. Dunn and Legerfelt (2005) suggested that COBOL cannot be written off as a dead language. Kizior, Carr, and Helpren (2005) noted that there were between 150 and 200 billion lines of COBOL code in business applications at the time of publishing their study and that several billions of lines of COBOL code were added annually to these applications. The availability of these large volume of code to maintain means that companies will keep using COBOL for a long time well into the future.
A question may be asked as to whether the companies that have large volume of COBOL code to maintain could move their applications to other platforms. Most studies conducted for this purpose found out that staying with COBOL is the most feasible approach for the company (KSVN & Choudhari, 2013, Matthiesen & Bjørn, 2015).

Suitability for Business Applications

COBOL started from the beginning on handling business applications. Examples of these applications include payroll, accounts receivable, accounts payable, point of sale, automatic tell machines (ATM) and many other similar applications. At the time of early use of COBOL in the late 50s and early 60s, no other programming was as suitable to handle these types of applications. The interesting point here is that still the same – no other programming language can handle these types of applications as efficiently as COBOL. Thus, these companies are staying with COBOL despite the many other issues that surface from using it.

Association with the Mainframe

In a study about the future of mainframe, Barnett (2005) noted that the mainframe is alive and well. Barnett also predicted that the future of the mainframe is bright as IBM continue to produce high end machines. The main point here is that on the mainframe, COBOL is the dominant programing language in terms of volume of code. So as long as the there is a demand for the mainframe, there is still a demand for COBOL programs. Thus, the need for COBOL programmers.
The original mainframe was developed by IBM in the 1960's and continued to grow throughout the years. After the turn of the century and the completion of the Y2K era, many predicted the demise of the mainframe (Sagers et. al (2013). Some predicted that the mainframe is going to end by 2005, others did not specify a specific time but predicted it will end (Fanelli, et al, 2016, Lewis, 1999). However, in 2006 IBM developed the Z9 mainframe featuring 64 bit processor and a 170 configurable core IBM Z14 model in 2017 (Craven, 2017), providing ample evidence that the mainframe is here to stay. Since COBOL is strongly associated with the mainframe, COBOL is here to stay.

The Modernization of Application to Cloud Computing
There is a move for most application to move toward cloud computing. This means dealing with existing programs as they move to the cloud. Among the code that companies have to deal with is the legacy code form old applications (Gholami et al., 2017). In other words, companies that want to move their applications to the cloud have to deal with their COBOL code as they migrate to the cloud (Maughan, 2017).
Companies have three options in dealing with the move to the cloud. First, to ditch out the old legacy applications and start new. Second is to replace the front end and keeping the back end at legacy applications. And third, is dealing with the legacy code that exist in a company's programs as they move gradually.

Barnett (2005) suggested that migration away from the mainframe is an excellent option for smaller companies but for larger companies, it will be too expensive to do so. Therefore, it appears that most large companies will stay with their legacy code (including COBOL) as they move to the cloud.

THE CASE OF A UNIVERSITY

Current Status
A department of Computer Science (CS) at a university located in Western of Pennsylvania teaches a course in data processing at the sophomore level that is dedicated to teaching COBOL programming language. The course is titled Applied Computer Programming (COSC 220) and emphasizes structured programming principles and techniques focusing on batch processing on an IBM mainframe. Some of the project topics include a simple reporting program, a two level control break report program, a simple sort program, and a sequential file update program. Interspersed throughout the semester, JCL (Job Control Language) statements are introduced and explained. This course exposes the students to a totally different platform and approach to programming than some of them have experienced before thereby greatly broadening thereby understanding of the computing field.
The Applied Computer Programming course is actually the second programming language course within the computer science degree programs. Students first take a Problem Solving & Structured Programming (COSC 110) course that introduces fundamental programming concepts using a subset of the Java programming language. So, students who take this course are familiar with programming syntax and have worked on developing programs. The Applied Computer Programming course builds upon this foundation introducing students to core algorithms for concepts such as group breaks and sort merge often used in programing business applications while at the same time solidifies an understanding of batch processing, rich data representations, and a different programming structure and syntax (yet leverages core understanding of English). As a result, students have a better grasp of computing and especially in understanding programming development in the context of business applications.

Against Teaching COBOL
The department is aware of the many factors that stand against teaching COBOL. Along with observations from Dunn and Lingerfelt (2005), some of the faculty view COBOL as outdated and believe there is a shrinking market base. Among the students, the comments are mixed. The difference in the approach and the structured layout appeals to some and are a constant point of confusion for others. Faculty periodically review comments from graduates and there is none so far against teaching COBOL as to warrant changing curriculum.
As far as the shrinking market demand, the department still believes there is a strong market demand now and this demand is expected to continue in the foreseeable future.

For Teaching COBOL
The department closely follows the market demand on graduates with certain programming languages and adjust their selection of programming language if needed. Lindoo (2014) performed a search for jobs using the "COBOL" as the keyword in an effort to show there is still a need for COBOL programmers. His search was conducted in January and then again in May of 2014. The table 1 shows his findings just for the United States, although his search also covered Europe, England, China, Australia, and Canada.

Table 1. COBOL Jobs in US Job Sites

Country	Site	Jobs 1/2014	Jobs 5/2014	% Increase
U.S.	Monster	165	196	19
U.S.	Indeed	1960	2170	11

A similar search on monster.com in May 2018, again using "COBOL" as a keyword, showed 1,000+ jobs for COBOL Programmer/Support Analyst, COBOL/.NET Programmer, COBOL/CICS Programmer. From indeed.com, it showed 1,083 job hits: COBOL Developer; Java Developer, COBOL Backend; Mainframe Developer; and, Senior Mainframe Programmer Analyst – just to name a few. From both sites, the job listings ranged from one to just a few days old, while others were 30+ days old.

Market Use of COBOL

As part of the curriculum, the degree program includes an internship course that is our flagship practicum experience that culminates a student's academic studies. In placing students at internship sites, we continue encounter demand for students with COBOL skills. Indeed, most of the major banks and insurance companies have internships (and jobs) using COBOL on mainframes. In talking with graduates who have interned and/or who have landed jobs with companies programming that use COBOL we have often heard that more (not less) time should be allocated to COBOL and mainframe processing to include further coverage of JCL, additional COBOL features, mainframe utilities, and even CICS (a mainframe transaction management system).

SUMMARY AND CONCLUSION

This paper is about teaching a course in COBOL. It contributed to the argument of the reasons that programs continue to teach COBOL in their courses despite the many counterpoints. It started by giving on overview of the factors that stand against teaching COBOL and then the factors that advocates site for their inclusion of COBOL programming language in their courses.

Based on our literature of review and the finding of this study, the department is inclined to continue teaching COBOL in the near future. The authors also intend to continue their research on the use of deployment of legacy applications to the cloud. We feel that this is a new area that is worth researching and that we may be able to find a new marketable area for people who have both skills: COBOL and cloud computing. We are hopeful that we will be able to develop at least a course in this area of cloud computing and legacy application, thus our intention is to research more on this topic in our next paper. Our paper is about legacy applications and the role they play in moving to the cloud platform.

REFERENCES

Ali, A. & Shubra, C. (2010). Dealing with Enrollment Decline at Computer Science Courses: A Case Study. *Issues in Informing Science and Information Technology*. *2010*(7), 210-224.

Ali, A. & Smith, D. (2014). A Debate over the Teaching of a Legacy Programming Language in an Information Technology (IT) Program. *Journal of Information Technology Education: Innovations in Practice, 2014*(13), 111-127. Retrieved from http://www.jite.org/documents/Vol13/JITEv13IIPp111-127Ali0773.pdf

Barnett, G. (2005). The future of the mainframe. *Ovum Europe Ltd.*, Retrieved July 5th, 2018 from http://jedi.informatik.uni-leipzig.de/de/VorlesMirror/es/Vorles/OvumReport.pdf

Bishop, J. & Horspoo, N. (2004). Developing Principles of GUI Programming Using Views. *Proceedings of the 35th SIGCSE technical symposium on Computer science education* (pp. 373-377), Norfolk, VA, USA.

Blansit, B. D. (2010). Object Oriented Programming: What is IT Talking About?. *Journal of Electronic Resources in Medical Libraries*, *7*(1), 90-97. doi:10.1080/15424060903585792

Craven, V. D. (2017, July). IBM z14 Features Improved Security, Insights and a Connected Ecosystem. *IBM Systems Magazine*. Retrieved from http://ibmsystemsmag.com/mainframe/trends/ibm-announcements/ibm-z14-trust/

Dunn, D. L., & Lingerfelt, D. F. (2005). Can Visual Basic replace COBOL? … and should it? *Journal of Computing Sciences in Colleges, 20*(4), 6-12.

Ehlert, A. & Schulte, C. (2010). Comparison of OOP First and OOP Later – First Results of Reading the Role of Comfort Level. *Proceedings of the 2010 Annual Conference on Innovation and Technology in Computer Science Education* (pp.108-112), Ankara, Turkey.

Fanelli, T. C., Simons, S. C., & Banerjee, S. (2016). A Systematic Framework for Modernizing Legacy Application Systems. *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)* (pp. 678-682). Osaka, Japan.

Gholami, M. F., Daneshgar, F., Beydoun, G., & Rabhi, F. (2017). Challenges in migrating legacy software systems to the cloud — an empirical study. *Information Systems*, *67*, 100-113. doi:10.1016/j.is.2017.03.008

Hajnal, Á. & Forgács, I. (2012). A demand-driven approach to slicing legacy COBOL systems. *Journal of Software: Evolution & Process*, *24*(1), 67-82. doi:10.1002/smr.533

IBM Corporation (2017). Master the Mainframe Contest. *IBM Corporation*. https://www.ibm.com/it-infrastructure/z/education/master-the-mainframe

Khadka, R., Belfrit V. Batlajery, Amir M. Saeidi, Slinger Jansen, & Jurriaan Hage (2014). How do professionals perceive legacy systems and software modernization?. *Proceedings of the 36th International Conference on Software Engineering* (pp. 36-47). Hyderabad, India.

Kizior, R. J., Carr, D., & Halpern, P. (2005). Does COBOL Have a Future?. *Proceedings of the Information Systems Education Conference 2000*, P. 126.

KVSN, S., & Choudhari, A. (2013). Legacy mainframe back-ends supporting new age enterprise applications: Can the elephant run with deers? *Proceedings of the 6th India Software Engineering Conference (ISEC)* (pp. 55-60). New Delhi, India.

Lewis, T. (1999). Mainframes Are Dead, Long Live Mainframes. *Computer*, *32*(8), 102-104.

Leutenegger, S. & Edgington, J. (2007). A games first approach to teaching introductory programming. *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education* (pp. 115-118). Covington, Kentucky, USA

Lindoo, E. (2014). Bringing COBOL back into the college IT curriculum. *Journal of Computing Sciences in Colleges*, *30*(2), 60-66.

Machuca, L. & Solarte, P. (2016). Improving Student Performance in a First Programming Course. *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 367-367). Arequipa, Peru.

Matthiesen, S. & Bjørn, P. (2015). Why Replacing Legacy Systems is So Hard in Global Software Development: An Information Infrastructure Perspective. *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing* (pp. 876-890). Vancouver, BC, Canada.

Maughan, J. (2017). *Cloud computing: can it play a role in the future of legacy applications*. A dissertation submitted to the University of Dublin in partial fulfilment of the requirements for the degree of MSc in Management of Information Systems.

Mitchell, R. L. (2006). Cobol: Not Dead Yet. *Computerworld*, *40*(41), 25-37.

Mitchell, R. L. & Keefe, M. (2012). The Cobol Brain Drain. *Computerworld*, *46*(10), 18-25.

Parker, K., R., Ottaway, Chang, J. & Chao, J. T. (2006). A Formal Language Selection Process for Introductory Programming Courses. *Journal of Information Technology Education*, *5*(1), 133-151.

Parker, K. R. & Davey, B. (2012) "The History of Computer Language Selection", in *Reflections on the History of Computing*, Springer Berlin Heidelberg, pp. 166-179.

Paulson, L. D. (2001). Mainframes, COBOL still popular. *IT Professional*, *3*(5), 12-14.

Phillips, B. K., Ryan, S., Harden, G., Guynes, C. S. & Windsor, J. (2013, May). Motivating students to acquire mainframe skills. *Proceedings of the 2013 Annual Conference on Computers and People Research* (pp. 73-78). Cincinnati, OH, USA.

Roggio, R. F., Comer, J. R. & Brauda, P. (2003). IS programs become accredited: COBOL in crisis. *Information Systems Education Journal*, *1*(15), 1-10.

Sagers, G., Ball, K., Hosack, B. Twitchell, T. & Wallace, D. (2013). The Mainframe is Dead! Long Live the Mainframe!. *AIS Transaction on Enterprise Systems*, *2013*(1), 4-10.

Smith, D. & Ali, A. (2014). Assessing Market Demand for Web Programming Languages/Technologies, *Issues in Information System*, *15*, 411-420.

Stern, N., N. & Stern, R., A. (2002). *COBOL for the 21st century 10th edition*. Wiley: Hoboken, NJ.

Van Gelder, A. (1977). Structured Programming in Cobol: An approach for Application Programmers. *Communications of the ACM*, *20*(1), 2-12. doi:10.1145/359367.359368

Wardlow, G. & Johnson, D. (1999). Level of teaching skills and interest in teaching improvement as perceived by faculty in a land-grant college of agriculture. *Journal of Agricultural Education*, *40*(4), 47-56.

Westfall, R. (2001) Technical opinion: Hello, world considered harmful. *Communication of the ACM*, *44*, 129-130.