The Future of Mainframe COBOL and the COBOL Programmer

Marshal Crawford – CEO Marble Computer, Inc.

Managing COBOL Code for Mainframe Applications has become a difficult task. The shortage of experienced mainframe programmers has left organizations with little choice but to offshore the maintenance of the application code with less experienced programmers or try to develop in-house COBOL programming talent.

This creates new problems in the maintenance of mainframe source code, the predominance of which is COBOL although Assembler and PL/1 as well as Java are routinely found on today's Mainframes.

IBM estimates that there is now around 350 billion lines of COBOL code being maintained throughout the world. These are mostly mainframe production applications that store and serve as much as 70% of the world's Enterprise Data (e-Week 5/30/14). With a dwindling supply of expertise and a retiring workforce what will happen to these Mainframe based applications?

Marble Computer believes that until there is a "Silver Bullet" solution to convert COBOL, Assembler and PL/1 to Java, the shortage of experienced mainframe programmers will cause organizations to:

- Induce retirees back to work, albeit at higher costs.
- Ship their mainframe code to offshore factories and live with the inherent performance issues and accept a higher rate of job failures (Abends).
- Count on offshore programmers who may be mainframe inexperienced and have limited knowledge of mainframe applications, data and JCL and where there definitely is a lack of good communications due to the programmers being half way around the globe.

While IBM and organizations like Compuware have pushed efforts to remove the coding and application maintenance functions off the mainframe, the absorption rate for software development environments like Rational, Topaz and other products continues to grow slowly. The requirement to add the Eclipse environment creates another level of support for Desktop management.

As for security, coding source on the mainframe is still the safest environment. Application theft is non-existent. Proprietary business logic, pricing and other key competition information is best kept off of a PC environment. ISPF works with mainframe security software that has been in use for generation of mainframe coders.

The preferred platform for developing and maintaining mainframe code is still ISPF. This is true whether your organization offshores or codes onsite or remote.

How the Problem Was Created

We have to go back some 50 years of computing history to see how the industry got to this point. Let's use April, 1965 as the starting point. The date of IBM's announcement of the 360 computing system, the forerunner of today's IBM mainframe products.

In 1965 mainframe computers were in use among many large organization, both commercial and governmental. There were several players in the computer hardware business. Those players included such names as IBM, Sperry Rand, Honeywell, RCA, GE, Burroughs, Control Data, NCR, XDS Systems and others.

Each hardware vendor supplied the hardware, the operating system, ancillary software and language compilers. The language compilers were proprietary in nature, Assembler, Autocoder, ALGOL, FORTRAN and of course COBOL. Later IBM added PL/1 (Programming Language 1) in order to combine the capabilities of COBOL and FORTRAN.

COBOL (Common Business Oriented Language) is an English like programming language that allowed for both commercial and some scientific use although FORTRAN was still the preferred language of choice for scientific use.

Below is an excerpt on COBOL from Wikipedia.

COBOL was designed in 1959 by the Conference on Data Systems Languages (CODASYL) and was partly based on previous programming language design work by Grace Hopper, commonly referred to as "the mother of COBOL". It was created as part of a US Department of Defense effort to create a portable programming language for data processing. Intended as a temporary stopgap, the Department of Defense promptly forced computer manufacturers to provide it, resulting in its widespread adoption. It was standardized in 1968 and has since been revised four times.

One of the strengths of COBOL is that as a programming language it is designed to be portable across different computing platforms from the hardware vendors. Each hardware vendor was supposed to supply a compiler that could take the COBOL source code as is, compile and execute on the vendors target computer. In reality COBOL written for one vender's compiler was almost, but not totally compatible across the hardware manufactures computing environment.

Business and most non-Federal Governmental agencies targeted their mainframe computing strategy around one manufacture's hardware with IBM the dominate share leader. IBM was so dominate that a new breed of computer manufactures called PCMs entered the scene a few years later. The Plug Compatible Manufacturers were able to supply all the hardware utilizing IBM operating systems and software as well as software from ISVs and the customer's applications.

The rush to take advantage of this fast growing computing environment led to a mass hiring and training of the COBOL programmer. The talent pool was comprised of newly graduated college students. However the demand worldwide for COBOL Programmers far exceed the supply so non college graduates that entered the workforce via COBOL training institutions added to the supply of talent.

The amount of COBOL lines of code that were being written, tested and put into production grew beyond estimates and there was still a lack of COBOL programmers. This led to a rise of new business that provided standard applications for business functions like Accounting, HR, and Manufacturing. These applications were also written in COBOL.

New programmer development tools such as Abend-Aid, File-Aid Xpediter, made it faster to write, and test code. Thus more COBOL code went into product year after year.

Compounding the development of COBOL lines of code was the constant change of the application's requirements due to governmental regulations or competition. Additionally, the lack of documentation of the application and programs became a big issue as the COBOL programmer moved from job to job due to short supply, high demand and even higher wages.

A Change of Computing Leadership

Some will argue that the change in computing leadership started with the Mini-Computer manufacturers, such as DEC (Digital Equipment Corporation), HP, WANG, Nixdorf, Burroughs and GE.

Some of these companies, like DEC, actually date back prior to the 360/System mainframe. However their rise in market penetration and market share came in the 1970's and had more to do with affordability, easier to use operating systems and interpretive programming languages like Basic.

Even IBM entered the minicomputer market with its System 3 product line that like the other mini-computer manufactures used an easier OS and language, RPG2.

While smaller companies and educational institutions flocked to the DEC product with its UNIX like Operation System VAX, other manufactures built MINI hardware around UNIX, a do it all operating environment developed by the Bell Labs division of AT&T.

But the MINIs were more of a departmental solution for most commercial and large governmental users. MINI's were best suited to a single application such as, word processing, data entry, forecasting and graphics. Scientific usage made up the bulk of DEC user community.

It is with the MINIs that we start to see an abundance of non-COBOL applications. A new and different programmer workforce built less on technical programming skills more on application functionality utilizing systems and languages that are more powerful and easier to use.

The Slide of the Mainframe

Ten years, almost to the date after IBM's System/360 announcement Bill Gates and Paul Allen founded the Microsoft Corporation. Their first product vision was building Basic Interpreters for small microprocessors like the Altair-8800. Microsoft entered the OS business in 1980 with its own version of UNIX, called Xenix.

Then in November 1980 Microsoft releases MS-DOS for IBM PCs (Personnel Computers) that the eventual slide of the importance of Mainframes has its genesis. IBM helped the slide by entering into the PC business and contracting with Microsoft for the now named PC-DOS operating system.

IBM further helped the Mainframe slide in 1984 by entering into a contract with Microsoft to develop a proprietary GUI OS for PCs called OS/2. OS/2 was designed for the business community, but this strategy was soon to fail and eventually OS/2 was taken off the market.

Microsoft took what it learned from developing OS/2 and released Windows for PC manufactures other than IBM on April 2, 1987. This gave rise to what many refer to as the WinTel alignment; Windows and Intel PC chips and processors.

A new breed of software vendors rapidly added to the Mainframe slide by providing new computer languages, infrastructure and database software. Applications that were easy to install and use covered the computer landscape. Graphical User Interfaces (GUIs) and the Mouse became the norm for WinTel based processors. All the while mainframe users were straddled with green bar paper and green screen CRTs.

Microsoft released the Office product line in 1990 and the end user community learned how to move the data off the mainframe into spreadsheets like Excel and Databases like Oracle. The Mainframe for the most part became a large data storage device processing large amounts of information at night in batch mode.

Many Mainframe online systems were ported to the new Open Systems Architecture. Many COBOL programmers decided to leave COBOL to opt for the more glamorous and higher paying coding languages like C, C++, JAVA and even Visual Basic.

The Mainframe decline is fast as we approached the end of the 20th century. The costs of operating Mainframes became considerably higher than the new Open Systems Architecture. Business oriented CIOs start looking at the Total Cost of Mainframe Ownership and strategically moved away from the mainframe for all new applications. Some CIO's began a plan to move entirely off the mainframe.

Thus we find an application fence is built around the Mainframe. The COBOL programmer becomes less valuable in the cost structure of programing skills, some exit the field and some begin to retire.

The Year 2K Crisis

The Y2K crisis which took off in early 1997 was a major shift in the history of the Mainframe and COBOL programmers. The crisis was created by industry pundits that predicted that many of the applications built in the mid-1960s to early 1970s would not function properly come January 1, 2000. The bulk of the predicted problems were due to the way dates were stored in most mainframe applications and purchased software.

Many IT managers were concerned about the time required to fix and test all the potential problems. Little documentation was available on these applications as well as the COBOL Programmers who wrote the applications. The words Legacy Application became an Industry term used for the mainframe applications.

Because of a lack of COBOL Programmers and the giant enormity of going back into these applications four distinct solutions became the norm. All four were another nail in the coffin for the Mainframe. They were:

1. Purchase the mission critical applications anew rather than remediate, albeit on a platform other than the Mainframe.
2. Offshore Coding factories were created, mostly in India, utilizing very newly trained COBOL Programmers.
3. Bring back early retirees who worked on these systems or those were adept at understanding the issues and could quickly learn the applications.
4. Use outside consultants from many small services organization and pay then enough to refocus their staffs on the Y2K solution.

There was a fifth solution not widely employed but available for production programs where the Source Code could not be found. Reverse Engineering – the creation of the COBOL code from Load Modules.

The Y2K crises seemed to never really happen. Many to this day believe it was a hoax. But result was twofold. The Mainframe gained more critics and Code Factories in India or other developing countries were now a part of the IT Management arsenal of options for maintaining COBOL programs.

Mainframe Customers Flee By the Thousands

At the height of the Mainframes popularity as a computing platform it is estimated that over 20,000 organization worldwide employed one or more mainframes to process their critical applications.

By 2015 the mainframe population has decreased to an estimated 425 commercial and governmental organizations employing several CPU across various data centers. Most large commercial organization outsource to a third party all or a large portion of their mainframe and application life cycle maintenance and operations.

There is currently, according to IBM, 350 Billion lines of COBOL in production worldwide, processing up to as much of 70% of all business transactions in one form or another. This might include data storage, batch processing or even online transactions. CICS the main transactional environment on IBM Mainframes processes up to 30 billion transaction a day. This account for more than $1 Trillion daily in business.

Most large commercial organization offshore the COBOL programming. However Federal and Local Governments are loath to send code overseas. This restriction occurs due to legal requirements or fear of a security breech.

Because of COBOL's very English like command structure much if not all an organization business or function logic is hard coded into the application's main processing programs.

COBOL programmers are paid less and so the attraction for a new breed of COBOL programmer is non-existent. Current Computer Science graduates prefer the more project oriented aspects of that curriculum which includes Data Base Administration, Systems Architecture and Data Center Management. While students are required to learn how to program the languages taught seem to be more of the JAVA and Visual Basic varieties.

In developing countries COBOL is a main computing language because the jobs for new graduates are in the Code Factories that dot the Far East and India. Code returned for production from these Code Factories is very commonly reported to be substandard and companies are willing to live with performance issues and program failures (Abends) which leads to higher processing and transactional costs. However this is offset by the cost of using onshore coders.

Conclusions

First let's be pragmatic, COBOL and the 350 Billion lines of code sitting on Mainframes is not going away over the next ten years or longer. Too many companies cannot get off the Mainframe no matter how badly they like to get from under this 50 year old technology.

COBOL Programmers in high standard of living countries are a dying breed. Countries with lower income levels will continue to churn out new COBOL programmers. Hired and trained for Code Factories these programmers will carry the maintenance load.

Larger companies like IBM, Compuware, CA Technologies, BMC and MicroFocus will continue to supply more advanced technology for dealing with COBOL and the Application Life Cycle.

Smaller companies like Marble Computer, Segus Inc. and RES Italy will have niche product lines that offer language advanced programmer productivity tools. These tools will automatically document the COBOL code and allow for less detailed knowledge about the code.

Eventually some company will come up with the "Silver Bullet" and COBOL will be converted to Java.

For now plan on the Code Factories, Re-hiring retired COBOL programmers and setting up COBOL Universities where possible. Given the scarcity of supply and the enormity of the demand we expect costs for maintain COBOL code will increase dramatically.