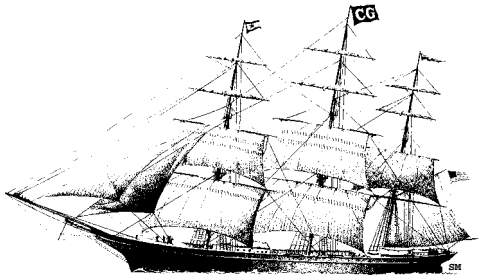
THE CLIPPER GROUP
Captain's Log™



Navigating Information Technology Horizons℠

Published Since 2001 Report #TCG2004036 April 9, 2004

The IT Civilization Needed a Language — IBM's FORTRAN Gave It Meaning

Analyst: Joe De Natale

Management Summary

The announcement of the IBM System/360 mainframe forty years ago was a momentous event in the world of IT. A scalable system that could handle the whole breadth of applications – hence the all-points-of-the-compass meaning of the 360. Granted, the scaling up had to be accomplished by replacement, but from the lowest to the most powerful, compatibility was maintained throughout the whole line of computers that spanned the range of 50:1 in processing power. A more exhaustive treatment of the System/360 anniversary is given in another *Captain's Log*.[1] This companion piece celebrates the 50th anniversary of the announcement by IBM of the FORTRAN compiler. That may not seem to momentous to those who were not there at the beginning, but as the saying goes, *You had to be there!*

Part 1 – In the Beginning

Back in the early 1950s, some wag invented the saying, *In the beginning was the word, and the word was 36 bits.* That is a pretty good starting definition of the *IBM 701*, which was announced in 1952. It was the first commercially-available internally-stored program computer from IBM. This was a tube-based computer and when I say *tube-based*, I mean tube-based. Not only was the circuitry based on vacuum-tube technology, but the memory was made up of 72 CRTs, which provided for an unheard of internal storage of 32K of 36-bit words. For the 701, the circuits were binary, not decimal-based, making it suitable for scientific calculations. Later a computer for commercial processing was announced, the *IBM 702*, whose circuitry was decimal-based. At a rental price of $15,000 per month (around $100,000 in today's dollars), the IBM 701 computer, with a processor speed measured at around three tenths of a MHz, was considered worth the price of 50 engineers or *calculators* that it would replace. These calculators were people who were performing primitive computing, where the processors were electromechanical calculators and the memory was accounting sheets. I knew those calculators first hand; I was one of them. So much for my B.S. in Mathematics!

Programming was primitive by today's standards, but when the Native Americans hollowed out logs to travel on water, it was easy to convince the tribe that it was a lot better than walking. Similarly, with those who

IN THIS ISSUE

[1] See *The Beginning of IT Civilization – IBM's System/360 Mainframe* in **The Clipper Group Navigator** dated April 7, 2004, at http://www.clipper.com/research/TCG2004028.pdf.

were now responsible for writing programs – whatever that meant – those primitive methods were an advance over what they had been doing.

Just think! One could, after debugging a program, add input to it, and run it on the computer in record time! You need another calculation? Just change the input and run it again. Processing times went from weeks and months to hours and days on this remarkable device. But there some hitches to this new and remarkable approach to personnel productivity. The IBM 701 did not have floating-point circuitry, a real hindrance for scientific calculations, especially when one considered the wide variety of engineering and scientific problems that customers wanted to solve. Floating-point calculations were made available through software, slowing calculations down but still better than what it had been. Topping it off was the fact that in designing programs, because of the limited amount of internal storage available, extreme care had to taken to organize the use of storage. That brought the process of "memory mapping" into being. Several layers of clear-plastic transparencies were marked off in different segments for the use of storage in the consecutive phases of computing. Data that had to be used in later calculations was stored on tapes or magnetic drums and retrieved as needed. On top of all that, programs had to be written and debugged in a substantially different way than we were used to, and there were few tools to assist in those efforts.

Part 2- Some Help is on the Way

The defense industry made up the bulk of the IBM 701 user community.[2] Whether it was aircraft fuselage, jet engine, or nuclear reactor design they all recognized the value of this new tool, but also realized that improvements had and could be made to improve productivity. Assemblers, many with these floating-point calculations simulated, were designed, written, and distributed

[2] By the end of the IBM 701's lifetime, which coincided with the availability of the IBM 704, 18 701s had been delivered.

Confessions of a FORTRAN Addict

It was late 1951, when fresh out of graduate school I was offered a job as a "calculator' at Pratt & Whiney Aircraft on a highly-secret project. I was not to know until I received my security clearance that it was a project to design a nuclear-powered aircraft engine. (Who would shoot one of these down?) My job was to perform hand calculations with the aid of an electro-mechanical-tabulating machine, solving many coupled differential equations.

On a trip to the Oak Ridge National Laboratory, I was exposed to the *IBM Card Programmed Calculator (CPC)*, which was being used to solve the same problems that I was working on. At the comparatively blinding rate of 150 calculations a minute (the card reader speed), run time could dramatically be cut down from weeks to days. The Research Laboratory at United Aircraft, the parent company of Pratt & Whitney, had several available. My calculations were converted to the CPC as programs. Now I was a *programmer*, a role that few could comprehend. When the IBM 701 was delivered to the Research Laboratory in 1953, I reprogrammed the CPC solution to work on it. The task was arduous, considering the complexity of the solution.

In 1957, I received a copy of the IBM FORTRAN specifications from Roy Nutt of United Aircraft and became enthralled by its possibilities. When FORTRAN became available, I reprogrammed once again, and promised myself it would be the last time, if I could help it. I could fulfill that promise, since now I was a *programming manager* and I could now set the programming standards. From the successors to the IBM 701 - the *IBM 704*, *7090, 7094 models I* and *II* to a "foreign" machine, the *Philco 2000*, and then to the *IBM System/360* and its successors - conversion was a snap, measured in weeks for several hundred FORTRAN programs.

FORTRAN, later *Fortran*, was and still is a powerful programming system. Although not designed for commercial use, I, like others, programmed a payroll system in Fortran. Later, I programmed a Braille conversion system for a blind programmer. More importantly, we used Fortran as the basis for the Apollo reentry vehicle design. Now, 50 years later, I am still impressed with that 1954 FORTRAN specification.

to the whole user community, free of charge. Yes, these were the days before software copyrights were established. Programmers would write their programs on paper program pads. They were then keypunched and assembled on the computer. The programmer still had to organize storage.

An attempt to simplify the programming with the introduction of a higher-level language called *Speedcode*, which was based on a three address, one line model (B = A + C, which means: retrieve the contents of A, add to it the contents of C and store the result in C). Looping and control was available on the same instruction line. Not a bad start but Speedcode was not a compiler, it was an emulator. Every Speedcode instruction line was decoded and then the operations were performed. A theoretical improvement had been made but run time was slowed down considerably and Speedcode, although used, did not catch on. It should be noted here that assemblers and Speedcode were the first example of virtualization in software. They abstracted the internal instruction set to a higher, logical level.

Part 3 - What was Needed

The users of the IBM 701, although impressed by what this new computer and the related assemblers and Speedcode, wanted more. The users both cooperated with and pressed IBM for new tools. There was a need to improve the time from problem conception to solution. The major steps from beginning to end in those days (not much different than today), were:

1. Problem analysis
2. Mathematical formulation
3. Systems analysis
4. Programming or coding
5. Debugging
6. Production

It was not unusual, and many times the practice, after the mathematical formulation was made, for scientist and engineers to submit the problem to a systems analyst. The analyst's responsibility was to organize the problem so that it could be coded for the computer. The coder became a sort of high priest who was among the privileged to enter the holy of holies of the computer room to submit programs for debugging. This was the closed shop concept that existed in many businesses. Despite the improvements made, barriers between problem origination and solution remained. There *had* to be some better way.

Part 4 - IBM's Response

The concept of a *compiler*, another virtualization tool, was considered. After research by a team of IBM personnel led by John Backus and bolstered by two people from the user community (the most notable was Roy Nutt of United Aircraft Corp. who later was one of the founders of Computer Sciences Corp) a more firm concept of a compiler was born. On November 10, 1954, a "Preliminary Report" was published by the Programming Research Group, Applied Science Division of IBM. This report of 29 pages outlined the *Specifications for The IBM Mathematical FORmula TRANslating System* (FORTRAN). By this time, the *IBM 704* had been announced, which had, among other improvements, a hardware floating-point instruction set, index registers, and magnetic core storage. On February 10, 1955, another report entitled *Proposed Additions and Modifications in the Specifications for the FORTRAN System* was published. Both documents were both remarkable in many ways, as remarkable as the announcement would be 10 years later of the IBM System/360 mainframe. It was a dramatic break from the past - a pioneering effort - with the possibilities of success or failure unknown, that was to permanently change what is now known as Information Technology.[3] Because of the mathematical nature of FORTRAN, scientific and engineering personnel could easily learn the language, eliminating much of the intermediate effort by people who had little or no understanding of the nature of the discipline.

[3] IT managers were now on their way up the corporate ladder from Tab Operations Supervisor, to Data Processing Manager, Director of Information Systems, and today, to CIO.

In retrospect, FORTRAN had limitations. Problems were coded as one entity. There was no subroutine capability, only what was described as an in-line Function expression. Although FORTRAN fulfilled its promise as a simpler and faster coding tool, it did not and could not live up to its ambitious and somewhat hyperbolic statement in the report that FORTRAN would virtually eliminate debugging. Despite that, FORTRAN would have the promise of fulfilling the goal that there would be substantial savings in time and dollars in problem solution. Because of its simplicity, FORTRAN would expand dramatically the number of people from different disciplines who could use the language. The "high priests", in many cases, could be replaced by scientists and engineers who could write their own programs.

Some features of the original FORTRAN would raise the eyebrows of people today who have little or no knowledge of that period. Two of FORTRAN's expressions may sound particularly quaint today. There was a PAUSE instruction where the computer would stop at that point and permit programmer intervention. The program would then continue on until another manual intervention or STOP. The STOP command was placed at points in the program where the programmer wanted to terminate the program. Imagine a Pause or Stop command on a z990 today!

Just two years later, IBM announced the availability of FORTRAN for the IBM 704. Early experiences with this new programming language were dramatic. It was truly easy to learn. Programs could be written by all levels of technical personnel. All of the specifications outlined in the original document were achieved, including several from the improvements outlined separately. Most significant was the EQUIVALENCE statement, which took the place of the storage mapping transparencies. Although most of the improvements were measured in program productivity, the greater result was that time to design and bring a product to the market was greatly reduced and scientific results were achieved in a much shorter time than previously.

Part 5 - FORTRAN Improvements

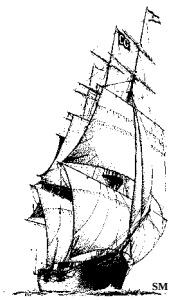
In the next few years several improvements to FORTRAN were made. Most significantly immediately afterwards was the SUBROUTINE ability, which allowed programs to be broken up into segments and debugged (for the most part) separately and then integrated as a total. A later version came with an operating system called IBSYS that could accept assembler written programs, FORTRAN and SUBROUTINE programs, along with data for execution of the compiled programs. By that time COBOL had been developed and was included in the support by IBSYS. One of the early goals of FORTRAN was to be able to compile programs whose quality was as good or better than the average programmer. Optimization features, *OPT 1* and *OPT 2,* were included in the FORTRAN IV release. From personal experience, in comparing the output from programmers in my staff with and without the OPT 2 feature, that claim was most often justified.[4]

Over the years, FORTRAN became available on a whole array of computing systems from PCs to open systems and to the successors of the earlier mainframes, up through today's z*Series*. Having gone through many upgrades in functionality over the last 50 years, *Fortran 2003,* currently under development, is the latest version. Fortran is still widely used globally to develop software to solve some of the most difficult technical problems. (For those interested in trivia, the word, software did not come into wide spread use until the early 1960s, although its first recorded use was in 1958.)

Part 6 - Still Around

Hats off to the giants of that era, especially to John Backus and his IBM team, to Roy Nutt of United Aircraft, and R. A. Hughes of the University of California, Livermore Radiation Labs.

Happy 50th to Fortran!



[4] Some may argue that I should have hired better programmers. If I had, the average would have gone up, but the claims for OPT 2 would still have been the same.

About The Clipper Group, Inc.

The Clipper Group, Inc., is an independent consulting firm specializing in acquisition decisions and strategic advice regarding complex, enterprise-class information technologies. Our team of industry professionals averages more than 25 years of real-world experience. A team of staff consultants augments our capabilities, with significant experience across a broad spectrum of applications and environments.

- ***The Clipper Group can be reached at 781-235-0085 and found on the web at www.clipper.com.***

About the Author

Joseph S. De Natale **is Director of Enterprise Systems Planning with The Clipper Group.** He brings more than forty years of experience in the data processing field with particular emphasis on systems management and application development on large-scale mainframes. Prior to joining The Clipper Group shortly after its founding, Mr. De Natale was an independent consultant where he provided expert opinion on data center management for civil cases. He later joined International Data Corporation (IDC), as a senior consultant and analyst, where he covered banking systems, data center management software, and large systems computers and storage. Formerly, Mr. De Natale spent eleven years at Citicorp Information Resources (CIR) as CIO of the Boston Data Center, where he managed the support of over 200 outsourcing contracts for thrift institutions. Earlier, he was MIS Director for the Lahey Clinic, and prior to that was a Project Manager for Computer Sciences Corporation, where he was involved with government outsourcing and applications contracts. Previously he was Director of AVCO Computer Services for fourteen years. At AVCO, he was responsible for all internal data processing, and initiated the marketing of computer services to commercial clients. Mr. De Natale began his career with Pratt & Whitney Aircraft as a programmer. During his career, Mr. De Natale was involved in the evaluation, installation and operation of large-scale mainframe systems and for the development of commercial, scientific and engineering application systems. He has also had successful experience in the marketing and operation of outsourcing contracts. Mr. De Natale attended Boston College's Undergraute and Grduate schools of Mathematics.

- ***Reach Joe De Natale via e-mail at denatale@clipper.com or at 781-235-0085 Ext. 24. (Please dial "1-24" when you hear the automated attendant.)***

Regarding Trademarks and Service Marks

The Clipper Group Navigator, **The Clipper Group Explorer**, **The Clipper Group Observer**, **The Clipper Group *Captain's Log*,** and *"clipper.com"* are trademarks of The Clipper Group, Inc., and the clipper ship drawings, *"Navigating Information Technology Horizons"*, and *"teraproductivity"* are service marks of The Clipper Group, Inc. The Clipper Group, Inc., reserves all rights regarding its trademarks and service marks. All other trademarks, etc., belong to their respective owners.

Disclosure

Officers and/or employees of The Clipper Group may own as individuals, directly or indirectly, shares in one or more companies discussed in this bulletin. Company policy prohibits any officer or employee from holding more than one percent of the outstanding shares of any company covered by The Clipper Group. The Clipper Group, Inc., has no such equity holdings.

Regarding the Information in this Issue

The Clipper Group believes the information included in this report to be accurate. Data has been received from a variety of sources, which we believe to be reliable, including manufacturers, distributors, or users of the products discussed herein. The Clipper Group, Inc., cannot be held responsible for any consequential damages resulting from the application of information or opinions contained in this report.