The First Commercial Computer Application at General Electric

By Burton Grad

Written: June, 2016

CHM Reference number: R0364.2017

Introduction

The General Electric Company (GE) was one of the industrial stars of the 1950s. It was certainly one of the largest companies in the United States with a dominant position in a wide range of electrical products ranging from small motors and electrical controls to house- sized large steam turbines and generators and electrical transformers. GE was not only the largest consumer electrical products manufacturer in large and small appliances, but also was a major producer of wire and cable, industrial diamonds, aircraft gas turbines, electronic communication products and nuclear reactors and materials.

GE had learned a number of lessons from the Second World War besides getting a strong foothold in a number of technologically advanced application areas. Among these lessons were the value of using analog computing facilities for scientific calculations and the enhanced use of punch card equipment for all kinds of business applications like accounting, manufacturing control and engineering support. It also latched on to the concept of Operations Research to apply advanced mathematical techniques to the solution of complex business problems. It created an Operations Research unit reporting to the highest levels in the company which attracted a number of outstanding mathematicians.

GE also had a reputation (well-deserved in my opinion) for being dedicated to training its managers in modern management processes and creating strong personal ties within its management team. But offsetting these positive attributes, GE also had a reputation (again well-deserved in my opinion) for anti-union activities. This was justified by its public relations department as necessary in order to eliminate its “communist-dominated” union (the United Electrical Workers) which could sabotage GE’s production facilities in any defense-related efforts (remember Korea and the “Cold War”). The corporate policy was called by its opponents “Boulwarism” named after the VP in charge of personnel practices, Lemuel Boulware. It was very much a “take-it-or-leave-it” attitude in negotiating labor contracts. This helped to make GE strongly focused on whatever could be done to make it less dependent on its union employees as well as to reduce costs to make it even more competitive in the marketplace.

So, automation (a dirty word at the time) for factory operations and later for office operations was a major goal of the company. This would make the company less subject to the impact of strikes and able to deal with known “fixed” expenses versus adjusting to ever higher variable costs. Effectively managing its productive resources (people, plants, assembly lines, parts manufacturing, purchasing and inventories, etc.) was a primary focus of GE management during the 1950s. Within this climate, GE encouraged the development of advanced tools for scientific, engineering and business applications.

Initial Experiences at GE

I had joined GE in 1949 as a Management Engineering graduate from Rensselaer Polytechnic Institute in Troy, NY and started in GE’s Management Training Program which included a lot of hands on assignments in testing equipment and analyzing business operations. After stints in Bridgeport, CT and York, PA, I was moved to Schenectady, NY to work at the then largest GE facility in the world. I was first exposed to the use of punch card equipment at GE's Large Steam Turbine Division (Building 273 in Schenectady) after having helped to test these enormous machines during the dead of night. This IBM punch card equipment was used to manage the work flow in the factory as well as to produce all of the cost accounting records and to perform engineering calculations.

I accepted my first permanent position at the Large Steam Turbine Division working in the Production Control department under Nels Coutant. One of my first assignments was to work on the factory floor in a dispatch cage to hand out the job tickets to the factory workers. This had to be carefully controlled so that the parts would be ready to put together the large steam turbines on schedule. At that time the work vouchers were on paper forms and one of the projects I worked on was to convert them to punch cards. This whetted my appetite for learning more about office automation and when I was offered an assignment at GE's corporate headquarters in New York City in 1954, I jumped at the chance. I was hired into the Production Control Services organization working for H. Ford Dickey. GE had organized its corporate staff to mirror the functional structure of its operating businesses: Engineering, Manufacturing, Accounting, Marketing and Sales, etc. Production Control Services was part of the Manufacturing Services organization.

Working with the Univac I

The Univac I had made a big splash in the 1952 presidential election by predicting the Eisenhower landslide even before the polls closed in California. It used statistical sampling techniques based on results from previous elections. The CBS network held back the predictions for quite a while (not believing the data), but finally called the results. The actual results came out very close to the initial predictions. Obviously, this was not just because of the power of the Univac I, but because some very savvy political analysts had constructed insightful models which had been accurately programmed and tested.

This was a wake-up call for many American businesses. If a computer could predict election results, why couldn't it forecast sales, lay out production schedules, simulate factory operations, perform "what-if" financial analyses and solve a whole range of engineering, scientific and operations research problems.

The first sale of a Univac I was to the US Census Bureau in 1953 to help process the 1950 census data. The first commercial sale of a Univac I was to General Electric in 1954 to use in its brand new Major Appliance Division plant in Louisville, Kentucky. GE had designed this state of the art factory with the most modern manufacturing facilities to produce washers and dryers, dishwashers and disposers, refrigerators and freezers and electric ovens and ranges. As part of its plan to automate its production facilities and make Louisville a showcase location, GE decided to use a computer not only to process payrolls and other accounting applications but also for manufacturing control and planning functions.

GE Corporate Accounting Services took primary responsibility for designing and programming the first payroll system, which was to be used initially by the Washer and Dryer Department. They assembled a large team of people from Accounting Services, from the Arthur Anderson accounting firm and additional people from Univac itself (which was a division of Remington Rand/Sperry Rand at that time).

GE Corporate Production Control Services took responsibility for designing and programming a manufacturing control system for the Dishwasher and Disposer Department. I was given this assignment and spent the next three months in Louisville (with my wife and 2 small children) developing these applications. I then spent three months back in NY City debugging the programs I had written (I'm not sure that it was even called debugging at that time). The programs were then installed in Louisville and were operational long before the payroll system was completed (possibly another argument in favor of Fred Brooks' theory as expressed much later in "The Mythical Man-Month" that putting on more people to do a job often increases the time required; it doesn't reduce it).

I learned a lot about myself and the nature of programming during that assignment:

1. I would never be a good programmer—I was fast but not careful enough.

2. Programming was incredibly tough without the computer assisting in defining the logic and in producing the actual machine code.

3. Computers were ideal for performing complex repetitive operations, but they depended entirely on the skills of the application designers and programmers to insure that the applications performed the business operations effectively and accurately.

In 1954 the Univac I had the following hardware capabilities: reasonably high calculation speed (for the time), slow internal memory (a mercury delay line), external tape drives (with metal tapes), printers, card readers and a control console. It had the following programming and data management facilities: none.

Programs were written directly in machine language with an operations instruction code (A for add, S for subtract, etc.) and an address (xxx) which was the actual word address in memory on which the operation was to be performed. The mnemonic instruction codes were pretty easy to remember, but the single instruction with one address at a time made for lengthy programs with many individual steps: adding a number to an internal register; multiplying a number by the number in the register and putting the results back in that register; taking the value of the register and putting it into an address for later use. It required three instructions just to multiply one number by another.

But the most painful part was managing the locations of the input and output information. The good news was that there were only 2000 locations (e.g. "words"). The bad news was that these were absolute addresses in the mercury delay storage device. So, any change in the records or fields usually meant changing the absolute addresses, hence reprogramming the application. Further, for efficiency, it was necessary to consider when each address would be available (there was a 200 millisecond lag before each address came around again for usage). So, based on the cycle time of each operation (add, multiply, etc.), one would try to position the addresses for inputs and outputs to optimize performance (improve throughput).

One of my special experiences in debugging (and correcting) the Dishwasher and Disposer manufacturing control programs was that I essentially had Remington Rand's NY City Univac I (which was used during the day for sales demonstrations) to myself from 6 P.M. to midnight each day (except for the computer operator); that meant that I didn't have to wait to run tests or to rerun programs or be delayed by anyone else's work. I could literally test, debug and correct in an online mode. The Univac I was also hooked up with speakers, and the operator had the machine playing classical music each evening.

Most remarkable (from my point of view) was that the programs really worked and were the first productive commercial applications to run on the GE Louisville Univac I when it was delivered in early 1955. However, through this experience, we began to understand that without better programming languages (relative addresses, multi-address instructions, if-then statements) and entry validation and test analysis tools, production programs would be slow to produce, difficult to maintain, costly to enhance and would operate relatively slowly.

The IBM 702 and Later Computers

In 1955, GE also acquired an IBM 702 computer to be installed at GE's Schenectady plant to run the payrolls for the departments located there (mostly electrical power equipment like large steam turbines and motor/generator sets). This project was managed by Jim Pontius who was located in Schenectady. The IBM 702 was a later design than the Univac I and had some advanced features like a vacuum tube internal memory with microsecond access speed and a relative addressing scheme. Unfortunately, IBM used a two-digit instruction numbering scheme, so it was not mnemonic and you had to remember that xx meant add and yy meant multiply. It was still a single address programming system. Pontius did an excellent job of planning and managing the IBM 702 payroll programming project and was able to complete the work in months, not years. This was certainly a major step forward and gave GE (and other corporation) executives more confidence that computer programming projects could be run like other facility construction projects with accurate task lists and definitions, realistic identification of skills needed and estimates of the time required for development as well as the ability to test the system "fully" before installing it.

With these experiences, GE became a hot bed of computer application development and IBM, with its 305 RAMAC and 650, 1401 and new 7000 series computers, became the leading force in computerization in GE while Univac relatively quickly lost its initial premier position.

As many of us know, to our chagrin, much of this executive confidence at GE and other companies regarding the manageability of program design and implementation turned out to be at least premature and, at worst, unrealistic (even to the present). In each generation of computers there have been academics and engineers, seers and prophets, structurers and organizers, scientists and promoters who have discovered the holy grail of how to define and implement complex systems and applications programs in a predictable and efficient manner. And so we came to depend upon subroutines, macros, reusable programs, structured programming, problem definition languages, application models, and component-based development; each was touted as the solutions to all of our problems.

But the greatest help for programming efficiency turned out to be the programming languages like FORTRAN and COBOL that were first developed in the late 1950s and early 1960s. These languages with their compilers did dramatically reduce the cost of programming, testing and the ongoing maintenance and enhancement. But neither they nor many of the other programming tools were ever really able to insure that the program designers or programmers could avoid logical errors or omission of critical decision factors.

Later, while still at General Electric, I came up with what we called Decision Tables as a means of structuring complex decision logic to try to solve this ongoing problem; it was intended to replace flow charts, decision trees and Boolean Algebra. But that’s another story. It never did become a commonly used tool nor did it successfully solve the problem of eliminating or dramatically reducing the creation of errant code and the need to spend endless hours and days debugging programs which unfortunately continues to this day (note the frequent updates that we get on our mobile device apps).