Learning about computers, programming, and computer system design circa 1963–1981

David Walden

July 30, 2011

In this personal history, I have chosen not to write comprehensively about the innovative technology developments I was close to in my computing career, but rather to focus on how I learned the technology of computing in the days when university computer science departments were still a new idea. Maybe this will bring back memories to others who lived through the same era of the computer industry.

1 Youth

I was born in 1942 in Longview, Washington, the son of public school teachers. My parents moved from Washington when I was a couple of months old so my father could take a high school teaching position in a small town in the Central Valley of California. When I was four years old, we again moved and, consequently, most of my youth was spent on the edge of the San Joaquin Delta in Pittsburg, California (where my father was a high school chemistry and physics teacher) or Antioch, California (where my mother was an elementary school teacher). The towns were six miles apart. The family moved from Pittsburg to Antioch as I was entering the eighth grade. I did well in math in high school. I enjoyed learning English composition in 10th grade English class. I also took three years of mechanical drawing (in the pre-computer days where we actually used T-squares and India ink). I really enjoyed mechanical drawing and decided to study architecture in college.

2 College

When I went away from home to college to study architecture at UC Berkeley, my father gave me only one piece of advice: "Have fun." I followed his advice to a fault; and, by the end of that first fall semester of 1960, I was on academic probation. Apparently all those hours of card playing instead of study meant I would not become an architect.

I moved back to my parents home and got myself signed up to attend Diablo Valley (junior) College in Pleasant Hill, CA, a few miles from home in Antioch. I declared myself to be a civil engineering major and continued the basic calculus-through-differential-equations sequence I had begun as an architecture major, as well as taking physics, chemistry, and civil engineering courses such as surveying, properties of materials, and statics.

By eschewing card playing at Diablo Valley College, my grades were quite good in the second half of my freshman year and throughout my sophomore year. By the fall term of

1962 my grades were good enough for me to matriculate to San Francisco State College as a junior, switching my major to structural engineering.

In the first half of my junior year at San Francisco State, I took another chemistry course and a couple of more math courses, but no actual engineering courses. (Of course, since starting a Berkeley two years before, I had also taken required and elective courses outside of science, engineering, and math.)

At San Francisco State I reverted to vast amounts of card playing and little study. Plus there were a lots of other non-academic things to do: drinking Red Mountain Burgundy (cheaper by the gallon), participating in civil rights marches, going to folk music concerts, visiting Palace Billiards, and so forth. My grades were not good.

By the second term of my junior year I concluded that I didn't really want to be a structural engineer and was faced with the question of how to continue having fun and still graduate in four years. Math was the only subject I'd been taking all along, so I visited Prof. Frank Sheehan in the math department and became a math major, despite having lower grades in math courses than I was getting in non-math courses. (I'm not sure why I knew to go to Frank Sheehan for help in changing majors because I'm unclear which courses I took which term; perhaps I had him for an introduction to a probability and statistics course — useful for card playing — that I had taken that first term of my junior year.)

3 Finding the computer

In the second term of my junior year (starting in early 1963) I took a numerical analysis course. The textbook was Milne's 1949 book *Numerical Calculus*. Although I often skipped class, the course was kind of interesting. For example, if memory serves, Prof. Bob Levit explained about positional number systems: since about the fourth grade I'd understood binary arithmetic from reading about solving the game of Nim (perhaps in George Gamow's book *One, Two, Three . . . Infinity*), but I'd never thought about the parallel between base-10 and base-2 arithmetic.

Early in the term of that numerical analysis course, we were given an assignment to be finished considerably later in the term — to do a numerical analysis project on the IBM 1620 computer shared by the science and engineering school and the business school. As the deadline for the computer project drew near, I eventually went and found the computer center; and the student assistants there showed me how to create punch cards on the 026 card punch machine, how to do a little programming in 1620 assembly language and Fortran, and how to assemble or compile and run a program. I think results were printed on the 407 tabulator. For people who don't know the 1620, it used base-10 arithmetic and had variable length instructions, and a 10 or 20 microsecond cycle time [`http://en.wikipedia.org/wiki/IBM_1620`].

I don't remember if I wrote my required program in assembly language or Fortran. I do remember is was some sort of Monte Carlo approach (I didn't know the word simulation then) to analyzing different parameters fed to a martingale (I was still pretty focused on games and gaming and was avoiding serious math). I also remember that I got a book from the college library that was on Monte Carlo methods or had a chapter on Monte Carlo methods by John Todd and Olga Taussky-Todd. This was the first computing reading I

did outside of looking at the machine and language manuals in the computer center, and the Todds were the first more or less computer people I ever heard of.

I'm not sure how many days it took me to understand how much I liked computing. Programming had the advantage that I didn't have to get the correct answer the first time. They even had a word for improving the program: debugging. I could debug and debug until it worked. I began spending a lot of each school day in the computer center. Now I had two things to occupy me instead of doing significant course work: computing during the day and card playing in the evenings. I was developing pretty good facility in both areas of study.

One of the student assistants in the computer center who helped me learn about programming the 1620 was Stan Mazor, another math student at San Francisco State. I was home in Antioch in the summer after my junior year, working in the paper mill, when someone from the computer center (perhaps Stan) telephoned me and told me that the computer center needed another student assistant the coming school year to replace a student assistant who was graduating; would I be interested in the job? It sounded good to me — making a few dollars an hour a few hours a week to spend time in the computer center (plus they gave me the computer center door key).

In my senior year, computing significantly cut into my card playing (on the side I did enough school work not to flunk out). I was captivated by computing. I visited Stacey's technical bookstore on lower Market Street and bought copies of Daniel McCracken's *A Guide to FORTRAN Programming* (Wiley, 1961) and Edward Feigenbaum and Julian Feldman's compendium called *Computers and Thought* (Edward McGraw-Hill Inc., 1963) and studied them thoroughly. McCracken's book was of practical use. Feigenbaum and Feldman's book was inspirational. I wanted to develop complex systems like their book described. I wanted to use the minimax method. I wanted to learn about list processing. I wanted to learn all about computing.

My adventures in computing that year took three forms. First, I learned a little about IPL-V. Prof. Ned Chapin and a few students (including Stan Mazor) were implementing an IPL-V interpreter, which got me interested enough to read the language manual (*IPL-V Programmers Reference Manual*, edited by Allen Newell, RAND Memorandum RM-3739 RC, June 1963) and to write away to university researcher for a listing of his contract bridge playing program written in IPL-V, which he graciously sent and I tried to figure out.

Second, I began running with a small clique of guys also excited by computers and computing. This group included Mel Jarenson, Larry Selmer, Mike Welker, and sometimes Stan Mazor until he left school to begin his illustrious career in Silicon Valley. We went to see the IBM 1401 at UC Medical Center where one of us (Mike Welker) worked part time or otherwise had an in. Uninvited we visited an IBM 7090 used by Chevron Oil on a high floor of a downtown office building. We attended Electrical Engineering Department seminars at UC Berkeley, e.g., an IBM presentation of its coming 360 computer, and a presentation of Space War by Steve Russell from the MIT AI Lab.

Third, Prof. Levit asked me to write a class scheduling program, I think for science, engineering, and math courses. I worked on this program for many hours in the second half of my senior year, reading the extensive literature on such scheduling, and got something rudimentary working.

About June 1, 1964, I graduated from San Francisco State. This took a little help

from Prof. Frank Williams of the business school. I had been helping Prof. Williams with his work in the computer lab, and he advised me on how to meet a particular graduation requirement without me having to take another course. A Bachelor of Arts degree with a major in math was my ticket to the white color workforce. (I had just kept taking math courses. Eventually they were useful for something.)

4 Getting a job

The bigger question during my senior year was what could I do after college. I clearly could not do math professionally (my overall grade point average was a B minus, and my math grade point average was some sort of C average). Really, the only thing I was plausibly qualified to apply for was a computer programming job. I went downtown and talked to an employment agency. It turned out that all the jobs they were trying to recruit for were business programming jobs, and it appeared that to do the kind of programming I'd read about in Feigenbaum and Feldman's book would require me (during that era) to live down the peninsula from San Francisco in what later became known as Silicon Valley. However, having grown up on the edge of the Central Valley of California, I had dreams of living in a big city and didn't want to go to a less urban city like Palo Alto.

So I sent inquiry letters and resumes to two institutions which in that era had an advertisement in every issue of *Scientific American*: NSA and MIT Lincoln Laboratory (I had the image that these were near Baltimore and Boston). As references, I gave Prof. Levit, Prof. Williams, and my fellow student Mike Welker. I think Prof. Levit was pleased at the idea that one of the graduates of his math department might get a job at one of those famous institutions (the math department at that time mostly trained people to teach high school math in addition to providing math courses for other departments in the college such as science, engineering, and actuarial science). Frank Williams was just a nice guy, who was enthusiastic himself and liked my enthusiasm. Being desperate for a favorable third reference I called on my friend Mike who at least had a part-time job in computing.

NSA responded first to my job inquiry and flew me to Washington, DC, where, along with other applicants, they put us up in a hotel and bused us to and from Ft. Meade for our interviews. It was the typical NSA interview that everyone has heard about: lie detector test, interview by a psychologist, and some sort of a math test, and of course talking to a technical person or two. I enjoyed the trip, was impressed by NSA, but I also now understood that NSA was not so close to Baltimore.

Back in San Francisco, Lincoln Lab finally called saying they would like to interview me. However, at that time computer programmers at Lincoln Lab were staff assistants, not full staff members (staff members were physicists, mathematicians, electrical engineers, etc.), and Lincoln Lab did not pay travel for staff assistants to come for interviews: did I want to pay my own way? I don't remember if I said I couldn't come if I had to pay my own way, or if I temporized somehow. In any case, a day later NSA called and invited me back for a follow-up interview. I called Lincoln Lab back and asked if they would fly me from Baltimore to Boston if I could get to Baltimore on my own. They said, "no," but said that if I could get to Boston on my own they would put me up in a Cambridge hotel overnight. I told them when I would arrive in Boston — the day after my second NSA interview.

In Cambridge, an MIT shuttle car picked me up at my hotel near Harvard Square and

took me to Lexington where Lincoln Lab has an entrance on Wood Street. (Once again, not so close to a big city.) There I met with people from probably three groups: the computer center (I think), another group which I can't remember at all, and Group 62 in the Space Communications Division. Lincoln Lab was impressive (the IBM 7094 in the main computer center was impressive, and someone took me to see the TX-2, or maybe it was the TX-0). I was interviewed in Group 62 by Frank Heart and Art Mathiasen. Frank talked to me and showed me around the Laboratory describing it in an exciting way. And I was excited back. Art was quieter but had an genial way about him, and he challenged me on Mike Welker's reference: "What is your connection to Mike Welker?" I admitted that I was hard up for a third reference who knew me as a programmer and that Mike was a fellow student with only a part-time job in computing.

Lincoln Lab's HR people told me that they would let me know; and, a while later, they called to offer me a job in Frank Heart's part of Group 62. I guess Frank had liked my enthusiasm or seen some future possibility based on the amount of time and intensity I'd dedicated to card playing and getting acquainted with computing.

I asked if Lincoln would pay for my move to Boston, but they told me that they didn't pay for moves of staff assistants. I thought for a moment about what I'd seen at Lincoln Lab (lots of computers, radar antennas on the roof, smart and fun-seeming people) and of the student scene in Cambridge, and told them I accepted and would be there shortly after the school year ended. I think I celebrated my 22nd birthday at home in early June and then flew to Boston where I quickly found an apartment and roommate a few blocks from Harvard Square in Cambridge. I bought a used car and began my daily commute between Cambridge and Lexington. My parents shipped several boxes of my stuff which arrived a few weeks later.

The year 1964 turned out to be good time to enter computing given my weak qualifications.

5 Lincoln Laboratory: becoming a journeyman programmer

When I arrived for my first day of work at Lincoln Laboratory, I was told that Will Crowther who I would be working with was on vacation; and I was given a copy of the manual for the Univac 1218 computer to study. My temporary office was in a lab some ways away from the offices of the rest of Frank Heart's subgroup. In the days while I waited for Will to arrive back, I mostly found my way around the lab — locations of the cafeteria, computer center, library, fast food grill.

I also read at the 1218 manual. The 1218 had 18-bit words, and most of the instructions seemed straight forward enough. However, I was puzzled by how the subroutine calls worked — didn't grasp it from the description in the manual. After a few days, Will arrived, came and found me, and asked how I was doing. I told him I was mystified by the subroutine call, and he expressed surprise without any hint of rebuke for my denseness. He said, "It's easy," and explained that when you did a Return Jump instruction to the memory location of the start of the subroutine, the address after the return jump instruction is stored in the first location of the subroutine, and the computer restarts execution at the second instruction of the subroutine (a return from the subroutine was done with an Indirect Jump through the location stored in the first word of the subroutine). Completely trivial,

but I didn't see it for days until Will explained it to me.

Will also briefed me on the system on which I would be working with him, The Lincoln Experimental Terminal (LET). The abstract from a Lincoln Laboratory report on the system (The Lincoln Experimental Terminal. Craig, J. W.; Crowther, W. R.; Drouilhet, P. R., Jr.; Harris ,J. N.; Heart, F. E.; 21 March 1967[1]), says

> Lincoln Laboratory has designed and constructed three satellite communications ground terminals (Lincoln Experimental Terminals) as part of its program in space communications. The first of these (LET-1) is a self-contained, transportable, X-band terminal in which the RF equipment, antenna and mount are on one vehicle, and the signal processing equipment, prime power and operating console are in a second van. The other two systems (LET-2 and LET-3) consist of signal processors identical to that of LET-1. They are similarly housed in vans and are used in conjunction with antennas and RF equipment associated with other terminals. The three systems were completed in 1965 and tested extensively since that time, using active satellite repeaters and the moon as a passive reflector. The function of the terminals is to demonstrate satellite communication in which multiple-access and anti-jam performance are emphasized. The use of vocoders for speech compression and an efficient modulation and coding system for signaling result in a low required satellite power per access. Bandspreading by pseudorandom frequency hopping provides the desired multiple-access and anti-jam capabilities.

Will was responsible for the design and implementation of the software for the system. It might have been the first system for which Will himself was the software leader. Art Mathiasen, who had interviewed me when I applied for the job at Lincoln, was responsible for the module that did the satellite obit calculations necessary to point the antennas. Will initially assigned me the job of writing the transcendental math subroutines. I headed down to the Lab library and found Volume 1 of Anthony Ralston and Herbert Wilf's *Mathematical Methods for Digital Computers* (Wiley, 1967) and read deeply the first chapter, talking over with Will what I was learning, e.g., about the speedy convergence of the Newton-Raphson Method for finding square roots. This also was my first introduction to both fixed point arithmetic and the distinction between one's- and two's-complement arithmetic.

Will also had me write a coordinate conversation routine (I think this might have been where I also had to implement a double precision multiply routine). I don't remember for what we needed to do coordinate conversion — from 3D geocentric coordinates in which the satellite equations were done to 3D surface-of-the-earth-antenna-position, from the 3D antenna position to azimuth and elevation for pointing the antenna, or for both of those. I do know that Will gave me the equations to figure out how to program.

Learning why such routines were needed in an antenna pointing system and the math involved in the routines themselves and how they could be implemented on the fixed-point 1218 was a fascinating. I loved it. I quickly had the transcendental math routines done and working, and Will said something complementary about me to Frank.

[1] `http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=AD0661577`

In those first days while I was working on the math routines, I still had my assigned space away from the rest of the group. But there was a little open area surrounded by the offices of the rest of the group, and I spent most of my time sitting in that open area working at little tiny table (maybe 15 x 18 inches) that happened to be there. I didn't consciously think about it; I just moved in with the rest of the group where I could see what was happening and easily get answers to questions. (Of course, today it is an important management principle that new employees should be geographically integrated among those they are working with, but all too often they still are stuck far away in the early months on a job. The new employee who cares about his success will find a way to be with the group.)

Shortly there was a rearrangement of a bunch of offices, and I was officially moved to be with the rest of the group and put in an office with two other staff assistants. Pairs of staff members shared an office. Threesomes of staff assistants shared an office. I also continued doing what Will told me to do.

Will wrote the master structure of the program, i.e., the interrupt organizations, and other software I don't remember, for the operational system. Whatever it was, he talked to me about what he was doing and began to teach me his thinking about programming:

- Minimize the size of the initialization routine by trying to structure the main loop so initialization happens almost automatically and not as a special case. More generally, avoid having code that doesn't run very often and thus might be prone to latent bugs.
- Always be concerned about performance in terms of speed: you have to build speed into the design from the beginning; it does not work to say you'll do it quick and dirty (and slow) initially and speed it up later. If the system is too slow, it will be unusable. If it is fast enough, it will be useable even if not fully refined.
- Keep in mind the possibility of pre-computing complicated and time-consuming calculations and doing table lookups in the running system.
- Use co-routines to interlink parts of the programs that synchronize with each other in irregular ways (the way the 1218's Return Jump subroutine call worked was perfect for co-routines).
- Learn the tricks of binary arithmetic: IOR, XOR, bit shifting, masking address fields for doing table lookups, e.g., in hash tables, etc.
- Write the whole flow chart for a big subsystem on one piece of paper. That way you can see the whole thing.
- Document what you are doing, however informally. The person who writes has disproportionate influence on what happens.

Will also had written a program for the 7094 that somehow shortened the edit, assemble, debug cycle. I can't remember what it was. Perhaps a 1218 assembler and interpreter so we could get our code mostly working without having to mess with the 1218 paper tape assembly system, or perhaps for developing the software before we had a 1218 available to us for debugging. Or perhaps we assembled on the 7094, and that somehow resulted in paper tapes which were patchable as we found bugs in the program (so we didn't have to

reassemble so often). I do remember patching paper tapes. (I also remember being very interested in the 7094's SOS operating system from SHARE and finding some literature to read about that. Later, I was vaguely aware that perhaps the first instance of CP/CMS was being brought up on the Lab's new IBM 360 computer.)

We did informal documentation as we went along: enough scraps of explanation and examples for the other members of the software team to use what we had written. The, somewhere along the line, we turned this documentation into Lab reports. I am listed a author or co-author of three of them.

- "Some LET System Math Subroutines," M.I.T. Lincoln Laboratory report MIT-LIN-62L-0061, April 26, 1965.
- "LET System Coordinate Conversion (CCCORV)," M.I.T. Lincoln Laboratory report MIT-LIN-62L-0068, June 15, 1965.
- "Univac 1218 Paper Tape Listing Program," M.I.T. Lincoln Laboratory report MIT-LIN-62L-0047, September 9, 1965.

Will also gave me the task of writing a routine to allow the system to read in a moon ephemeris paper tape derived from data obtained from the Naval Observatory and to use that to produce antenna pointing data based on the date and time. My understanding was that the obit equations for the moon were too complicated to calculate within the system. (More about this in a few paragraphs.)

Not so long into all of this, Frank Heart moved me into a two-person office with Will. I guess it was clear that we were working well together and that I was enthusiastically diving into the thick of things. I shared that office with Will for the rest of my time at Lincoln Lab, and my education by him as a computer programmer went on hourly. If I was working on something and Will sensed I was struggling, he'd ask me to tell him about it; and then he would teach me what I needed to know. Occasionally I'd help him too, such as in those instances where he'd have me look at a piece of his code where he couldn't see the bug and a fresh pair of eyes might.

As the parts of the software system came together in a 1218 in one of the group's laboratory rooms, I followed Will around full time as he did the system integration—eventually as *we* did the system integration. Will treated me like his partner in this system effort despite the great disparity in our skill levels. And sometimes I worked on system debugging alone (I tended to get to work in the morning before Will).

I remember one very-hard-to-solve problem. The system crashed occasionally with long periods of time between crashes. Studying the code, putting in breakpoints, etc., didn't find the bug. But Will had us keep a record of the crashes so we were able to calculate the average number of hours between crashes. We then calculated the frequency with which each interrupt routine would run on average. From this we were able to say which pair of interrupt routines ran with appropriate frequencies such that, if there was an interrupt bug between them (e.g., one didn't properly save a register the other was using), it would cause the system crashes with the average frequency we were seeing. Focusing on this pair of routines, the interrupt bug quickly was spotted. Of course this statistical approach to finding a problem came from Will, but now I also had a new tool in my debugging kit.

Eventually we installed our software in the 1218 in the mobil van, and my memory is that it mostly worked without much additional debugging. Our testing in the lab had been sufficiently thorough. There was one problem however. After the system had been running for a few months and Will and I had moved on to new things, I got a phone call saying the the moon pointing part of the system had stopped working. It was on about June 1. I went to the van, confirmed the problem, and got out my listing of the code. I had been studying the listing for a couple of hours not seeing anything, when I looked again at the table of cumulative days of the month I had created to convert the day of a month input into the system into a day of the year for the ephemeris lookup. As I looked at the table, I once again recited the famous memory device: "Thirty days has September, April, May and November; all the rest have..." Fortunately, I was mumbling this semi out loud, and Will heard me and was able to point out that I had had a memory glitch, and that of course June, not May, had thirty days. A quick tweak of the table and reassembly of the system paper tape, and the problem was solved.

I don't remember where we were in the development of the LET system, but after six months at Lincoln Lab, Frank Heart came to me and told me that they were raising my salary a bit. I had been happy to go to work at the Lab for what they offered me, I was enjoying myself and learning a lot, and this was completely unexpected. Years later when I had become a manager, this lesson stuck with me. You don't have to stick to annual reviews, standard guidelines for treating people equally, etc. If someone is performing unexpectedly well, the time to tell them you appreciate them is now.

Other members of our group included Ben Gold and Charlie Radar who were working on the digitation, compression, and resynthesizing of speech (vocoder technology). This was the early days of digital signal processing and they were educating themselves about it. After the LET system was working Will began to drift into the speech area and I followed. He had some ideas about somehow using Hadamard matrices. Related to this (or related to something related to this), Will asked me to write a program to calculate eigenvalues and eigenvectors. Looking back, I'm not sure why we didn't use an existing routine to do this; I suppose one may not have existed already, but that seems unlikely. In any case, I didn't think to ask this question and set off studying the literature of numerical analysis and eigenvalues and eigenvectors. I made many trips to the Lab and MIT libraries and read all I could find. Three of the books I remember studying were James Wilkinson's *The Algebraic Eigenvalue Problem* (Oxford University Press, 1965), Alston Householder's *The Theory of Matrices in Numerical Analysis* (Blaisdell, 1965), and Marshall Pease's *Methods of Matrix Algebra* (Academic Press, 1965). As I studied, I talked over the options with Will and found what seemed like the optimal method for Will's use.

From the beginning of my time at Lincoln Lab, I had not worked simply as a coder, only implementing an algorithm given to me by someone else. Rather, I took the problem, perhaps including some equations, and I studied it thoroughly, reading everything I could get my hands on and talking things over with Will (who had been a physics major at MIT and was strong in math), and then implemented what seemed to be a good approach to the problem. This study didn't slow me up, as I was basically working or studying computers day and night. I've never thought until now about whether Will was consciously using this approach as part of my education or if he was just going with my flow.

I got the eigenvalue and eigenvector program working, and through this project learned

in detail lots that I had skimmed by or avoided studying in my college math career. In the end, I wrote up this effort as a Laboratory Technical Report:

- "The Givens-Householder Method of Finding Eigenvalues and Eigenvectors of Real Symmetric Matrices," M.I.T. Lincoln Laboratory Technical Report No. 1967-51.

Technical Reports were more formal than the reports we wrote just to document parts of a project. There was some peer review, and one of the assistant group leaders called me in and talked to me about my draft of the report and asked me to do a little more reading and to provide a little bit better explanation of some of the math. I can't remember if I took it upon myself to write this report, or if I did some basic documentation and someone higher in the group suggested that I write it up more formally. Either way, the Lab was treating me like a contributing scientist and allowing me the important new experience of writing a formal report.

As I followed Will's drift into the speech processing area, I had some (in retrospect trivial) ideas of my own about using correlation methods (delta-coding, or something) to do speech compression. I wrote some code for the 1218 and tried out my approach, and it produced results of some sort. I felt like it was doing something interesting. But in retrospect the interesting part of this exercise is that the people around me took me (if not it) seriously. I was asked to give a presentation on "your work" to the Joint Advisory Committee (JAC) that reviewed Lab activities periodically. The presentation was not long — 10 or 15 minutes — but Charlie Radar coached me on its preparation and presentation. He had me write out the presentation in advance, and then memorize it. He made the point that if I knew the presentation by heart, I would be able to give it spontaneously even if I did not use the exact memorized words.

I gave the presentation, and it went acceptably well. Charlie's lesson is one I have used the rest of my life: over prepare for public presentations. With a presentation thoroughly in mind, with much more detail than I may need in mind, I am able to give presentations that sound relatively spontaneous, remain coherent, and fit within the scheduled time. Again, I don't know what the group management was doing in having me give the JAC presentation, but I suspect that they were taking explicit steps to develop my capabilities.

Somewhere in the time period I was working on the 1218 in the speech area, it was decided that four of us would write decent software for it. Ralph Alter, who had come to the Lab from the MIT AI Lab, wrote a version of TECO (Dan Murphy's paper Tape Editing and Correcting Program). By this time we had Model 33 TTYs with paper tape punches and readers. Will wrote a version of Alan Kotok's DDT debugging program (including some other capabilities) which he called Utility. Chuck Neissen and I wrote a version of Peter Sampson's macro assembler: Chuck wrote the basic assembler and I wrote the macro-processor. In other words, we copied the suite of software that had developed around MIT for the PDP-1. I was listed as the author of two reports on this effort:

- "UNIVAC 1219 Macro Assembler — Operating Instructions" C.W. Neissen and D.C. Walden, M.I.T. Lincoln Laboratory report MIT-LIN-62L-0097, September 26, 1966.

- "Utility - An I/O and Debugging Package for the Univac 1218," D.C. Walden, M.I.T. Lincoln Laboratory report MITLIN-62L-0052, February 24, 1965. I don't remember

why I am listed at the author of this report. Maybe I wrote up what Will had implemented.

The macro processor was the first recursive program I ever wrote, and it is was a pretty full facility macro processor. Thinking through how to write a recursive macro processor was itself an education. I think this was also the first time I wrote code for managing a symbol table. This was also the first time I had seen map commands, i.e., apply a macro over a list of arguments; Sampson called his commands IRP and IRPC, if I remember correctly. I still have my program source code listing and could look it up. I'm not sure if I had already read Christopher Strachey's paper on the GPM macro processor in *The Computer Journal* by then or not.

Sometime shortly after I joined Lincoln Lab, I joined the ACM. I read the issues of the *Communications of the ACM* avidly. I also made trips to the MIT libraries to check out books, for instance, volumes of the *Annual Review in Automatic Programming* series. A while after I arrived at the Lab, Dr. Henry Fractman returned to the Lab from his posting at the Lab's facilities in the Marshall Islands. Henry had an almost complete set of the CACM from a year or so after its inception until my subscription started in 1964. He gave me all his old issues, and I studied that back issues avidly too, reading about the algorithms, the languages and the computers, and learning the names of the people in the field, particularly focusing on the names of people around Boston. I couldn't get over the beauty Warshall's algorithm for transitive closure. Somewhere along the line I also subscribed the *The Computer Journal* of the British Computer Society.

We had visitors from MIT to our group all the time. Sometimes they were jointly working on projects in the group, and sometimes they just came to talk to us about something (well, not really to talk to me, but to talk to someone of their own stature in the group). I don't remember if Jack Wozencraft's lecture on Curry Functions was just to our group or to a larger set of people, but I do remember that Christopher Strachey had been visiting MIT, Jack had talked to Strachey about Curry Functions, and Jack apparently had been asked by someone to pass on what he knew about Curry Functions. I never quite saw the point, but Will got fairly excited about them for a few days. I did buy Curry's book to try to understand better.

In 1966 Frank Heart left Lincoln Laboratory to go to Bolt Beranek and Newman Inc. (BBN). As my supervisor at the Lab, Frank had treated me like a first rate citizen. Our ID badges indicated whether we were staff members or staff assistants, but Frank escorted me into a couple of staff seminars to which staff assistants were not invited. I already mentioned the early pay raise during my first year and placing me in a two-person office with Will; I don't know how much else of the career development things that happened for me happened because of Frank's support. Frank also helped me get into MIT as a part-time Master's student (more about that in the next section).

A while after Frank left, I was asked by Walt Morrow (of another division, who later became Laboratory director) to write a Fortran program for him that would calculate beat frequencies among known frequencies of radio transmission. He gave me the math, and I quickly wrote the program. This was the first time at the Lab that I'd served more like a coder (although I did learn about beat frequencies).

I'm not fully sure why I became unsatisfied after Frank left, but it began to matter

more to me that I was not a staff member. Certainly there was no longer an explicit digital computing subgroup in Group 62 after Frank left. I asked Irwin Lebow, our deputy group leader (or perhaps by then he was group leader), about becoming a staff member. He suggested that this could happen after I received the MS I was studying for part time at MIT.

MIT Lincoln Laboratory had been a real education in computing for me, and it fundamentally had been a joy to be there. However, in September 1967 I followed Frank Heart to BBN.

6 The MIT community

When I moved to Cambridge in 1964, I found a guy needing a roommate in an apartment on Hancock Street. While the apartment was closer to Harvard Square, it was only about a mile from MIT. After about nine months, I moved to an apartment a five minute walk from the edge of the MIT campus. In the 47 years since then, the average distance of my Boston area house or apartment has never been more than 7 miles from MIT and averaged of about three miles from MIT (for 12 years I lived on the Boston bank of the Charles River looking directly across the river at the MIT Great Dome). Although I went to BBN in 1967, MIT has remained a big part of my life.

In my earliest months in Cambridge I wandering the halls of MIT finding where computing was happening and finding information on computing:

- the CTSS system in Building 26
- Project Mac and the AI Lab in Tech Square
- the Barker and Hayden libraries and their math and computing books and journals
- the Lewis Music library (not computing but still an enjoyable resource)
- the MIT Coop bookstore, then in a small building at the corner of Massachusetts Ave. and Amherst St.
- etc.

I had the claim of being an MIT employee and a Lincoln Lab badge if anyone called me on going into places where I wasn't invited. (I also found the Aiken Center Lab at Harvard.)

While at Lincoln Lab, I went on a few MIT Outing Club weekends with Will Crowther (rock climbing or skiing). However in the fall of 1964, while wandering the halls of MIT, I stumbled on the MIT Gilbert and Sullivan Society rehearsing for "HMS Pinafore" in the Walker Memorial building. From then on, my MIT-oriented social life revolved around amateur theater — the Gilbert and Sullivan Society, the MIT Community Players, and the MIT Classical Musical Society which I co-founded. I met my wife to be when she tried out for the Classical Musical Society's initial production, "Guys and Doll." She was from outside of New York City and was working at the Harvard Medical School, and any thought I had about eventually returning to the West Coast was no longer relevant. (Even at a technically oriented university environment such as MIT, a social life is available.)

A ways into my time at Lincoln Lab, I got the idea that I should take some computer science courses part time at MIT. I talked to Frank Heart about it and, although MIT didn't have provision for part-time grad students, Frank talked to Herb Teager in the Electrical Engineering Department (Course 6 as it is universally known at MIT where departments are known by numbers rather than names — the math department, for instance, is Course 12); and in 1966 I started taking computer science courses part time at MIT. I would go to work at the Lab in Lexington in the morning, take the hourly MIT-Lincoln Lab shuttle car to MIT in Cambridge in time for class on class days, take the shuttle back to the Lab after class, and then go home to Cambridge in the evening. For some reason Paul Rovner was going back and forth between Lincoln Lab and MIT in the shuttle cars in this same time period, and I got to know him pretty well.

The first course I took (in the Spring of 1966 — probably on some sort of probationary basis) was Herb Teager's 6.543 Advance Computational Systems Seminar. I don't remember anything about lectures that Prof. Teager may have given. I do remember that much of the course was taken up by small groups (three people per team?) studying (e.g., by doing a literature search) a significant type of computational system, writing a report, and presenting the report to the class. I think the team on which I participated investigated weather computations which also took us into the realm of parallel processing. Doing our report was certainly educational for me: I had never thought about large scale numerical computing or parallel processing before. I also enjoyed hearing the other reports, but didn't learn as much from them as I did from working on my team's study and report.

In the Fall of 1966 I took Theoretical Models for Computation (6.253) taught by Peter Denning, then a PhD candidate. This was a course that I believe Prof. Jack Dennis originally created (with David Kuck), and it was the hardest and most interesting course I took at MIT (in general MIT graduate courses were not very hard, but this was an undergraduate course). The course covered state machines, regular expressions, Post productions, Turing machines, the halting problem, deterministic and non-deterministic computation, various types of grammars, and so on. In addition to voluminous class notes that Peter handed out (which years later became the book *Machines, Languages, and Computation* by Denning, Dennis and Qualitz), I also dipped extensively into outside materials: Halmos's book on naive set theory, Trakhtenbrot's pamphlet on algorithms and automatic computing machines, Martin Davis's book on computability and Turing machines, and papers by researchers such as Shelia Griebach and Seymour Ginsburg in the *Journal of the ACM*. I mentioned that the course was hard.

- It had hard but interesting homework assignments.
- I completely botched a mid-term exam.
- The final exam I remember as being a *many* day take home exam involving (perhaps among other things) a kind of theoretical machine we had not studied in the term.

Fortunately, I was taking the course on a pass/fail basis, and I apparently did not do badly enough to fail. While the topic was theoretical models, in fact I used what I learned in this course over and over in the rest of my computing career as I implemented parsers, programs with state variables, and so on. I also took out of this course a vastly increased ability to read the literature of computing. In retrospect, this was the best course I had in all

of undergraduate and graduate school. And I have kept in touch with Peter Denning ever since as he has progressed through his renowned computing career.

I took three more courses at MIT, in the Spring 1967, Fall 1967, and Spring 1968 terms. I have a record of the course numbers and their non-specific names (e.g., "Studies in Computer Science"), but I don't remember which professors and content went with which course names and numbers. (In the Spring of 1967 I also left MIT Lincoln Laboratory and moved to Bolt Beranek and Newman Inc., continuing my part-time MIT study.)

One course was taught by Prof. David Martin who was later a well known professor at UCLA. I remember that there was a guest lecture by Bill Poduska, but I remember little else about the content of the course.

Another course was taught by Prof. Jerry Saltzer and someone else and was on principles of computer system design. Saltzer and other members of the computer science faculty were implementing Multics at the time (and also, I believe, modernizing the computer science curriculum); and they were teaching the ideas, such as late binding, memory segmentation, etc., that came out of the ambitious design of Multics. (Faculty members Joe Weizenbaum and Bob Fenichel audited this class.)

The final course was taught by Prof. Martin Richards, temporarily at MIT from his U.K. university (he had been a student of Strachey, I believe). Prof. Richards had designed the BCPL language and implemented it while at MIT circa 1967. In addition to teaching us about BCPL and how it was implemented, Prof. Richards also had each of us study another language processing system, write a report about it, and present the report to the class. My report was on Val Schorre's Meta syntax-oriented compiler. This was another course where what I learned I used throughout the rest of my computing career. (Also BCPL had an era — prior to the existence of the C language — when it was perhaps the most popular programming language in the early ARPANET community.)

Somewhere along the line, I took the qualifying test for a Master's degree in computer science, and I guess I passed it. The rest of my MIT career was spent researching and drafting a thesis about an extensible macro processor. Prof. Bob Graham was my thesis supervisor, and I learned a lot about macro processors and extensible languages. I began to write the program implementing the macro processor I designed, but I didn't finish the implementation (personal and BBN business issues took over my time), didn't get a Master's degree, and eventually withdrew from study at MIT. I remain, however, a proud alum.

7 BBN

I was to start work at Bolt Beranek and Newman on a Monday morning in September of 1967. I was so anxious to get my hands on the PDP-1d time-sharing system people in my BBN division had developed that I talked the BBN guard into letting me into BBN's 20 Moulton St. building (in the BBN complex of buildings in northwest Cambridge) on Saturday morning before the Monday morning on which I was to start work. I spent the day incrementally reimplementing and debugging my Fortran program for the Givens-Householder Method in BBN's Stringcomp interpretative language using the Model 33 Teletype terminal in my new office. Experiencing time sharing was wonderful.

I can't remember exactly what I was assigned to do during my early months at BBN. I remember that on-and-off over my next three years there I worked on several design or

consulting projects.

At one point, Bob Kahn and I were assigned by Danny Bobrow to look into SDC's TDMS (data management system). I think we were to make some sort of recommendation about it to the Air Force. We visited an Air Force base near Washington, DC, where we heard a presentation on the system, visited SDC in Santa Monica (hosted by Clark Weissman, who also showed us their time-sharing system), and wrote our report. This study gave me some beginning education on issues in database management system design (I also studied what BBN was doing involving inverted lists and efficient data packing on disks for Massachusetts General Hospital).

At another point Alex McKenzie and I looked into the capabilities of various minicomputers of the time for a project the division had in the life sciences area. That result in time fed into the choice of the computer for the ARPANET project.

Also, Paul Castleman and I spent some time working with Dr. Bill Raub of the National Institutes of Health in formulating a project that would integrate programming, data management, and data visualization for medicinal chemists and pharmacologists. The system was eventually named the Prophet system. I was already studying macro processors and extensible languages for my thesis work at MIT, and I did a literature search, wrote a report ("An Informal Note and Some Readings on Extensible Languages," BBN accession number 001.642 W162I, 1968, 389 pages), and proposed an extensible language for Prophet. I also wrote a position paper explaining why the system we were proposing should be based on an extensible programming language so the chemists and pharmacologists could have chemistry-related data types and operations in their programming language in addition to the typical algebra-like constructs and types of languages like Fortran. To show the feasibility of this, I obtained the Fortran listing of James Bell's PhD thesis on the PROTEUS extensible language and spent a few days transliterating it line-by-line into PDP-10 assembly language. We probably claimed some sort of milestone with this implementation, and it may have helped us promote the project in some way with Dr. Raub and the researchers he served; however, the implementation never really worked. Fred Webb was asked to take over maintenance of my implementation (as I moved deeper into BBN's ARPANET project). Fred doesn't know how much, if any, of what I did got preserved. The implementation philosophy changed completely at a very early point in his involvement and at that time he may well have started again from scratch. Webb's version of a language for Prophet was Prophet was named PARSEC.

I also did lots of what we then called "programming hacks" (implementations of interesting programs done without permission from management). In particular, I was interested in programming languages and reimplemented several systems:

- Strachey's GPM (general purpose macro-processor) transcribed from an issue of *The Computer Journal* into PDP-1 assembly language.

- Calvin Mooers's TRAC reimplemented in PDP-1 assembly language (Peter Deutsch had already implemented the system once). As part of this effort, I visited Calvin Mooers in his office nearby Cambridge (stuffed with boxes of his archives) and visited Peter Deutsch at his parents house in Cambridge.

- Nicholas Wirth's Euler implemented in Lisp on BBN's SDS-940 system (using Warren

Teitleman's DWIM system). Bernie Cosell and I shared the job with one of us writing the parser and the other writing the interpreter.

Reimplementing an existing program, even by direct transcription, seemed to me to be an excellent way to study the work of the "old masters." I also studied Algol and the work of the Algol 68 committee.

As a result of my study of languages, parsing, interpreting, and compiling, at some point Frank Heart had me teach a session or two on this subject in the Northeastern University extension course on computing that he taught. I also taught several sessions on BBN's Telcomp language in BBN's commercial Program for Advanced Study. Everyone knows that preparing to teach about a subject is a good way to organize and solidify one's knowledge of the subject.

During my time at BBN (both before and after my time at Norsk Data — see Section 9), I began subscriptions to additional ACM journals: the *Journal of the ACM*, *Computing Reviews*, and *Computing Surveys*. I also kept buying books — Saul Rosen's *Programming Systems and Languages*, Gordon Bell and Allen Newell's *Computer Structures: Readings and Examples*, Knuth's three volumes of *The Art of Computer Progamming*, and many others particularly in the area of languages, operating system, and computer design. I also subscribed to the *IBM Systems Journal*. Eventually I developed a personal library of *many* books on various aspects of computing. Considerably later, I also subscribed to the IEEE Computer Society's *Computer* magazine.

Also, in about 1969, Danny Bobrow gave me a review to do for the ACM's *Computing Reviews*. This was my first outside publication, I think, and led over the next few years to a total of nine reviews in that journal.

Sometime in 1968 several of us became involved in an effort to bid on an upcoming ARPA Request for Quotation to build the ARPANET packet switches. My consequent education in and involvement with data communications is the topic of the next section.

8 ARPANET

The early ARPANET story in general is well known (e.g., see Katie Hafner and Matthew Lyon's book *Where Wizards Stay Up Late*, Simon and Schuster 1996), and I will focus on my part in it. Probably sometime in relatively early 1968, BBN began to pull together a small team of people to prepare to bid on an expected Request for Quotation (RFQ) from the Advanced Research Projects Agency of the Department of Defense for a contractor to develop the Interface Message Processor (i.e., packet-switch) for the first four nodes of the ARPANET. Bob Kahn (and perhaps others at BBN) had been aware of the coming RFQ and was an communications theory expert, so naturally he was on the team. Frank Heart was an respected development manager and he was named team leader. Severo Ornstein brought hardware design and development expertise to the team, I brought software development expertise, and a few other BBN people participated in less central ways. We began creating the overall system design and the hardware and software designs for the IMP system. I was learning about packet switching, cyclic redundancy checks, and so on as we conversed and designed. Nonetheless, I was experienced with real-time systems implementation and did a competent job of sketching out the structure of a program:

- to handle inter-IMP communications in and out
- to handle host/IMP communications in and out
- to handle routing update calculations and other more or less background calculations run between running the communications routines run off interrupts from the hardware communications interfaces

We did quite a lot of design in advance of the arrival of the RFQ.

At some point, Severo, Frank, and I decided it was time to encourage Will Crowther to leave Lincoln Lab and to join us in bidding on the IMP design. This was with my strong encouragement as I was not completely confident of my ability to develop the software for this high visibility project alone. Will did come to work with us.

I'm not sure if we already had the RFQ in hand when Will arrived or if it was about to arrive, but Will jumped into the system and software design; and he and I worked together as smoothly as we had at Lincoln Lab. By the time we sent in the proposal in response to the RFQ, our team had a very detailed hardware, software, and overall system design; actual code for the key inner loops for which Will and I counted the instructions cycles; and we thus had calculated accurate delay and throughput numbers for packet processing performance. We won the competition and began funded work on January 1, 1969.

We refined the system design in the early days of 1969 and made some further changes as we began the implementation. I'm not sure exactly when various people were added to the project (before or after our proposal went to ARPA, or after we had won the job). In any case, Ben Barker joined the team to help Severo with the hardware, and Bernie Cosell joined the team to help Will and me with the software. On the software side of things, Will implemented the inter-IMP, source-IMP-to-destination-IMP, and dynamic routing code, I implemented the IMP/Host code and some of the other background routines, and Bernie Cosell implemented software to help us develop the program on the time-shared PDP-1 (rather than on the Honeywell 516 for which we were implementing the system) and various debugging tools integrated into the running IMP system. Everyone contributed ideas, and we helped each other develop ideas. I do think that I had the initial idea of a hardware addition to the computer such that the program could set an interrupt at the appropriate place level in the interrupt priority hierarchy. I also think I had the initial idea that expanding the IMP's capacity to handle multiple host computers could be extended to handle what we called "fake hosts," routines within the IMP that were communicated with as if they were hosts on the network which allow cross-network debugging, tracing, statistics reporting, and so on. The three of us worked together and individually to integrate and software and debug the operational system, and eventually we all could work with any part of the code.

The first IMP was delivered to UCLA at about Labor Day in 1969. Ben Barker and Truett Thatch went with that first delivery. I went out to UCLA shortly after the machine was installed and running. I also became, in effect, the software system maintainer. People phoned me when they wanted something done with the IMP software, including at home where I kept a program listing by my telephone. I went with the fourth IMP delivery to the University of Utah, and then on at least one occasion (before we got the fifth IMP at BBN and had the ability to do new program releases over the network) I carried new-release paper

tapes to all four of the initial West Coast IMP sites. Also, after the first four IMPs were running and communicating on the West Coast, I went with Bob Kahn to UCLA where we ran a series of well known experiments that showed that the source-IMP-to-destination-IMP algorithm we had implemented had lockup problems that required Will and Bob to design and Will to implement a new algorithm.

Our contract was extended after 1969, and BBN continued delivering IMPs at a rate of about one a month for the next few years. I was in the thick of network operation and software maintenance (as were others) until I left BBN in September 1970 to go work in Norway.

9 Norsk Data

My wife and I moved to Norway (with our 3 month old son) because we had a desire to live in Europe and Norsk Data replied to the job inquiry I sent, interviewed me, and offered me a job.

When we got to Oslo in September 1970, I immediately went with two other Norsk Data people to Italy to prepare to bid on a communications system there. (We didn't get the job.)

Next I began work on the design of a packet-switching network for the Norwegian Air Force, where I pretty much copied our BBN ARPANET IMP design, including changes (e.g., to the inter-IMP acknowledgment scheme) that happened after I left BBN but that Bernie or Will told me about at long distance. This implementation of the second packet switching network is described in a paper Nils Liaaen and I wrote many years later: "Remembering the LFK Network," *IEEE Annals of Computing*, Volume 24, Number 3, July–September 2002, pp. 79-81.

Probably in early 1971, ND and BBN bid together to build a packet-switching network for the Scandinavian Airlines System, but we did not win the job. Still, it was fun to have Frank Heart, Severo Ornstein, and Hawley Rising from BBN visit Oslo.

Later I got thinking about how the Nord-1 computer should have a time-sharing system, and wrote a memo to my boss Rolf Skår sketching how such a system might be implemented. Finally, I was the person who put Bo Lewendal in touch with Norsk Data and encouraged ND to hire Bo. Bo did join ND and implemented a state-of-the-art (for mini-computers at the time) time-sharing system for the Nord-1 (using what he had learned working on time-sharing system development with Butler Lampson at al. at Berkeley Computer Corporation). Bo's system led to great ND success over a period of years. (I like to think my memo had some influence on ND's willingness to let Bo develop his system.)

Although my wife and I had intended to stay in Norway for several years, by mid 1971 I was missing some of the broader computer activity going on at BBN, asked Frank Heart if I could come back, Frank hired me back, and I restarted work at BBN in September 1971.

10 BBN again

Back at BBN I rejoined BBN's (by now expanded) team of people working on the ARPANET. Upon returning, I think my first job (for which I volunteered) was to specify and implement

the IMP's Very Distant Host Protocol. After a little while, Frank and Will decided that I should lead the IMP software effort with Will nominally working for me. Will had little interest in managing things, and I found some gratification in keeping track of things and beginning to interact with our ARPA customer and our networking colleagues at other institutions. Of course, I was well aware that me being Will's nominal boss didn't change how much smarter he is than I am, and we continued to get along just fine. On the technical side, I continued programming things occasionally, but more and more I documented things (writing was easy for me, others didn't like to do it so much, and it needed to be done). I also began giving presentations to outsiders on what we were doing (I was becoming known as an expert in this field of packet-switching that I had entered without any prior knowledge: it helped to enter the field when I was one of the original workers in the field). My CV (`http://www.walden-family.com/dave/archive/cv4.pdf`) has a fairly complete list of papers, reports, and presentations of mine from that era.

The first few years after I returned to BBN I was involved in various protocol developments. I was probably the first person to draw the famous ARPANET protocol layering figure. I helped Bernie Cosell invent the Telnet Will/Won't/Do/Don't mechanism for negotiated options and drafted the addition to the Telnet protocol spec; see "Development of TELNET's Negotiated Options," Bernard Cosell and David Walden, *IEEE Annals of the History of Computing*, Volume 25, Number 2, April–June 2003, pp. 80-82. I contributed to the discussions that led Vint Cerf and Bob Kahn's codification of TCP for the Internet and was involved in the first internetworking experiments. And so forth.

In time I became responsible for more and more of BBN's networking activities (which included the Pluribus parallel processor packet-switch) and interacted with groups around the world trying to learn about packet switching (including Louis Pouzin's group implementing the Cyclades system) as well as interfacing to our government and commercial customers.

One activity that I think was particularly noteworthy was that John McQuillan and I taught a graduate seminar one year at Harvard which was probably the first university course on packet switching. Some of our students became quite famous in the computer field, for example, Eric Roberts (Stanford faculty) and Guy Steele. Guy told me one time that our description of the ARPANET (distance vector) routing algorithm was the first parallel program he studied.

At some point Frank Heart appointed me to be co-assistant (with Paul Castleman) division director. Recruiting good additional people was an important part of the job as was helping deal with overall division finances, pay raises, etc.

At another point Frank Heart recommended me for the Air Force Scientific Advisory Board, and I was on that board for eight years (and also on the Electronic Systems Division Scientific Advisory Board for part of that time). I was a member of the Electronics Panel of the Air Force SAB, and in that role I saw a lot of complicated computer systems and processes including for radar emitter location and recognition, multi-level security, management of major software acquisitions, and the upgrade of the F-111 avionics system.

I did keep reading and studying technically on my own time. For instance, while another group of BBN people was developing the TENEX time-sharing system, I worked to keep up with what they were doing and also to learn what they were learning about other time-sharing systems such as Multics.

I stayed working (and studying) in a like manner (a little technical, and quite technically diverse, but mostly managerial) until 1980 when I pushed for a BBN activity independent of Frank Heart's division.

In the last year before I left the division, I bought BBN's first personal computer, an Apple II, on which to run Visicalc for division financial forecasting. I like to think that I thus began BBN's move into the era of personal computing.

11 Infomail

John McQuillan and I have been talking about doing something outside Frank Heart's division. We talked to BBN's top corporate management, and they agreed to let us write a business plan which ended up being for a multi-platform email system to be sold to the commercial world running on companies' mainframe and mini-computes.

Our activity was named BBN Information Management Corporation (BBN IMC), and our product was named Infomail. John handled the marketing and customer service activities, and I handled the development and overall management and finance activities (this is where I really learned how double entry bookkeeping works). I hired the development staff for Infomail, did the initial product design and specification (including an early desktop and file cabinet model), and personally implemented the operational backup system which was also used for moving the email database from one release to the next of the system. We implemented the system in Rarfor (Rational Fortran) which permitted us to code in a more C- or BCPL-like language that output Fortran for compilation on the several different computer and operating system platforms on which our system ran.

The system worked quite well and we made a number of sales, but not enough to be a commercial success. In time, the system was mostly used throughout the Defense Data Network for which BBN had become the implementor and operator.

In the meantime, I left BBN IMC and moved to BBN Computer Corporation, BBNCC (later BBN Communications Corporation), to serve as general manager. Eventually BBN IMC was shut down, John McQuillan left BBN to become a famous consultant, and the Infomail activity was merged into BBNCC.

12 Departure into higher management

Becoming general manager of BBNCC in 1981, I basically stopped thinking about technical things. I gave my extensive library of computer books to Roxbury Latin School where my son was attending high school. I threw away my class notes from MIT course 6.253, my essentially complete draft MIT thesis, and my files from the early days of the ARPANET. I thought I was now going to be a manager and didn't need these technical materials anymore. Within a few years, I regretted not keeping all this stuff. I left general management after about 10 years and wanted to think about technical things again, or at least to write about things we had done, and the original materials were gone.

I served as general manager of BBNCC for a couple of years and then became general manager for about eight years of BBN's traditional research and development business (which was 1,000 people by the time I left it). In this latter role, I became familiar with

a broad range of technologies (some of which are described in two special issues of the *IEEE Annals of the History of Computing* on Bolt Beranek and Newman Inc.: Vol. 27, No. 2, 2005, and Vol. 28, No. 2, 2006). As general manager of BBN's R&D business, I had the advantage of being able to ask someone to give me a couple of hour personal lecture on any BBN area of technology that I wanted to better understand. (In this role, I also had collegial relationships with Al McLaughlin who ran a division at Lincoln Lab, Mike Dertouzos and Al Vezza of MIT's Laboratory for Computer Science, and Keith Uncapher of USC ISI. I had occasional show-and-tell exchange visits or discussions with each of these major computer science research directors. I also participated in various ARPA contractor meetings where I heard about many other computer R&D efforts.)

However, during this period I also did one more bit of programming. Bob Thomas and Will Crowther were trying out some of Will's ideas for utilizing the Butterfly parallel processor. To better understand what they were doing, I also programmed two or three algorithms using their system for programming parallelization (I think maybe a matrix multiply, a sort, and maybe one other algorithm).

After eight years, I left BBN's R&D activity to become BBN's corporate quality officer. Two years later, I became general manager of BBNCC again in what ended up being a two-year effort to spin off some of its activities and to integrate other parts back into BBN's R&D activity. When that was done, I left BBN, taught part-time at MIT Sloan School of Management and with the Center for Quality of Management for several years, and then stopped paid work.

13 Retirement

In my retirement years, I have mostly done writing and publishing — about management practice, computer history, and computer typesetting. I have also done various programming projects for organizations for which I volunteer.

Like anyone with some serious familiarity with computers, in retirement I help my family and friends with their personal computer system admin needs. I also learned to create and manage websites (my own and for various organizations) and began writing (eventually fairly elaborate) Perl programs to automate the generation of the websites (`http://walden-family.com/texland/tb100berry.pdf`). In other words, after almost 15 years away from technical work, in my retirement I became a computer programmer again — part time.

Early after my retirement from BBN, I gave up using Microsoft Word for significant writing projects and adopted TEX/LATEX for these projects (I had already used command-based systems such as Jerry Saltzer's RUNOFF system, John Barnaby's WordStar system, and the UNIX nroff/troff system, and I had admired Don Knuth from afar for decades. Thus, I adopted the command-based TEX and its derivatives to write about management practice (`http://www.walden-family.com/public/mybooks/`) and computer history (`http://www.walden-family.com/dave/#ref-ieee-activities`), and this led to also writing about typesetting (`http://www.walden-family.com/texland/`). I have also written combinations of TEX and Perl programs to automate various aspects of the typesetting process and conversions into TEX or HTML markup (`http://www.walden-family.com/texland/tb95berry-interviews.pdf`). As an important part of the TEX world is

now using the Lua programming language, I will probably begin to use that at some point.

Writing this account was stimulated by computing history retirement activities with the IEEE Computer Society (`http://walden-family.com/ieee/`).

I look forward to what the next years will bring as my education in the use of computers continues. However, one thing has never changes. Despite various programming disciplines that have been popular during my era of computing (the move from assembly language to high level languages such as BCPL or C, avoidance of goto's, structured programming, object-oriented programming, etc.), in any high level language I still program as if I was writing Fortran in 1964, e.g., `if X eq Y, goto Label A`.

14 An aside on writing

I had enjoyed writing in high school and did better on the SAT verbal part than on the math part (on which I did quite well). However, before college I had heard many people complain about term papers during their college years; and thus when I went to college I avoided any course that had a term paper.

I then wrote technical reports while working at Lincoln Laboratory and remembered that I enjoyed writing. At BBN I continued writing, more and more. I helped write reports and proposals to potential clients. Even when documentation was not required, I got in the habit of writing as a way to sort out my thinking on a subject and sometimes to help me explain to others an idea I had and to solicit feedback.

For example, in the summer of 1970 I thought about and wrote a couple of RFCs (RFC 61 and RFC 62 superseding 61) on "A System for Interprocess Communication in a Resource Sharing Computer Network" that was different than the Host-Host protocol that was developed by the Network Working Group. (It was also early thinking related to what have become known as datagrams. These ideas were published in the CACM in April 1971 in a paper of the same name — my first refereed publication.)

I have been writing extensively ever since restarting writing at Lincoln Laboratory. When my then young son was asked at one point asked his father did, he said, "He is a typist": I was always at my computer terminal either programming or writing.

Of course, despite the stupidity of avoiding courses with term papers in college which I might have enjoyed, my life has probably gone better because of my drift into math and then computing (where my willingness to write was a big plus). If I'd been taking those college courses with writing assignments, I might have drifted into a liberal arts career instead and missed the explosion of the computing field.

Corrected and slightly updated 2011-12-11