An Interview with

PETER C. PATTON

OH 325

Conducted by Philip L. Frana

On

30 August 2001

Minneapolis, MN 55413

Charles Babbage Institute
Center for the History of Information Processing
University of Minnesota, Minneapolis

Peter C. Patton Interview

30 August 2001

Abstract

In this oral history Peter Patton, Chief Technology Officer at Lawson Software in St. Paul, MN, and former Director of Academic Computing at the University of Minnesota, talks about his education at Harvard, his involvement in the development of PLATO courses in the humanities, and his perspectives on the software industry and patenting. Patton also shares his experience with IBM 650, CDC 6600, and Cray installations on several university campuses, his role in Project Safeguard, and his design of the Hennepin Justice Information System.

Oral History:

Peter C. Patton

This is an Oral History with Peter C. Patton conducted by Philip L. Frana on August 30th 2000, in Minneapolis, Minnesota, for the Charles Babbage Institute Center for the History of Information Processing. This interview is conducted under auspices of the National Science Foundations Software History Project.

Frana: Well, Peter, I just want to start with some background information on your education and the first question that I had was about your Harvard days. Your first major was Sanskrit and then you changed over to Engineering and Applied Physics. How did that come to pass?

Patton: Before I went to Harvard I had as a high school student taken an interest in philosophy and particularly in logic, and as a matter of fact I did my high school 11th grade term paper on a computer developed by Ramon Lull in the eleventh century, who was a Spanish monk, to do logical propositions.

I made one with construction paper and turned that in with this term paper, which was a hundred and eight pages. I remember the grade I got. I got an A+ and a comment "This term paper was ten times too large." (Laughter) But, so I had that interest and when I got to Harvard, the first semester I took a lot of philosophy courses and I was disappointed to learn that Aristotle's logic had not been pursued in the West. But I learned from Willard Van Orman Quine, a professor of logic in the philosophy department, when I took his course spring semester, that Alexander the Great, had taken Aristotelian scholars to India with him, just like Napoleon took his scholars to Egypt, and they had taught the

Indian pundits, Aristotle's syllogisms, twenty-five syllogisms, sixteen were valid and nine were invalid. Nothing further was done with that in the West, however, the Indians developed it into what became called known as Navya-Nayya Logic and it was used to develop the Upanishads. So I asked Professor Quine how can I learn this logic and he said go see Professor Ingalls who is the Sanskrit Professor here and he just wrote the only book in English on this topic and he's an expert. So I went to see Ingalls, who hadn't had a Sanskrit major in ten years, so he held me hostage to major in Sanskrit. And so I majored in Sanskrit. About halfway through my sophomore year, I discovered computing and saw computers as a possible intellectual prosthesis for me and Sanskrit. I changed my major to engineering and applied physics instead of computing and I'll never forget the look on Ingalls' face when I went in and told him I was dropping my Sanskrit major. He was of course terribly disappointed, and, that was a hundred percent loss for him, (laughter) and but the funny thing is his son, Daniel Henry Holmes Ingalls Jr., also went into computer technology. So it was a proleptic experience for him to see that. And he died about six months ago and when I found out about it I got on the Internet and I found the bookstore that was selling his forty thousand volume library and I bought a seven volume edition of the Ramayana from Princeton with his bookplates on it and that's my memorial in my own sense to him.

Frana: Now you mentioned to me that there was a specific event of the encoding of the Book of Job.

Patton: Yes, one evening about midnight I was coming home from the Sanskrit Library, which was a big room in Widener Library at Harvard. And when you come down the front steps and turn right you go by this building which was then called the Harvard Computation Laboratory; it was like a fishbowl and inside there were three computers: Mark I, Univac I, and, Mark IV. And so I went in. I

saw a priest in there and I don't remember now whether he was Catholic, he was Catholic priest, because he was wearing a cassock, and the contrast of this struck me. Here is a person wearing a uniform of the magistrate of ancient Rome working on this then very modern technology. I stood there at midnight looking at this brightly lit building. I couldn't resist, my curiosity overwhelmed me. So I went to the door and knocked and it was locked at that time and the priest came to the door and said, yes, what did you want, and he knew I was an undergrad because I had a green book bag over my shoulder and I said, well I told him the reason for my curiosity. And he said, 'What do you do?' He said, 'Well I am working on the Book of Job.' And I said, 'On the computer?' And he says, 'Yes. Do you know the Book of Job?' I said, 'Yes, of course', because as a thirteen year old I had read Ray Bradbury's story "The Firemen" which later he turned into a full novel called Fahrenheit 451 and I thought, well if there is ever a book burning I want to memorize a book so I tried to memorize the Book of Job. So I started quoting the Book of Job to him in Hebrew with an English translation after each line and he says, 'Come in, Come in!' (Laughter) And so he spent the evening, well the morning I guess by then, showing me how he had taken the Book of Job and he had coded it in the Hebrew letters like 11, 12, 13, 14 and 15, two decimal digits, and he had written a program and he was trying to classify the tri-literal Semitic stems in Hebrew, into Elamite, Chaldee, Hebrew, Arabic stems because at that time, this would have been '55, the approach for the Book of Job which has probably more hapax legomena, once said words, than any book in the Bible, except perhaps the Song of Solomon. But certainly the high point is the Divine utterance in the Book of Job, chapter 38-41; since these words didn't occur elsewhere in the Bible and the ancient Hebrew language is defined by the five thousand words, in the Hebrew Bible, what we the Christians call the Old Testament, hapax legomena had to be from some other language. And clearly the Book of Job is the oldest book in the Bible. Hebrew tradition says it was written down by Moses they thought that he

learned it in the Sinai Desert in the forty years he was there and he wrote it before he wrote the five books of Moses. So this priest was trying to find the roots for sources, the lexigraphical sources, in the Book of Job. Now we understand it very differently. I just had a graduate student do his Ph.D. at the University of Texas Arlington, on the unity and symmetry in the Book of Job and we now know that all those strange once said words in Job are Hebrew. But nonetheless it was really research at that time and it fascinated me that here was a machine that could help me with the problems I was interested in using for Sanskrit. So I went around to the Harvard Computational Laboratory the next day and the guy I talked to was Ken Iverson.

Frana: Sure.

Patton: And he recruited me eagerly for his course the next term entitled "Programming." And it was one of the best courses I've ever taken in my life. He was a graduate student at that time.

Frana: At that time?

Patton: Yes.

Frana: I read through the class notes that you took for his applied mathematics class and was amazed at how precisely he'd laid out the class, it shows through in your notes.

Patton: Yes, he would walk up to the board at the beginning of each class period and he would write out an outline on the board and the outline, first item in the outline was always numbered zero. And

then the second item was 1. And then under 1, there would be 1 point zero and he would write it in his tiny, tiny neat handwriting. And down the left side of the board he would write about eighteen inches, the things he was going to cover that day, and he kept exactly on that material.

Frana: Did this turn into a textbook? I don't recall seeing it.

Patton: No, it didn't. But clearly, years later when I was reading his book, Programming Language, it was very familiar, he didn't use that term, in this course at Harvard, and of course, the way programming was taught then, computers didn't have that abstract symbol manipulation capability. But Ken Iverson taught us programming as not numerical computation, but a manipulation of abstract symbols. So for him to develop abstract symbols for transposing a matrix or something, you know, was a natural step in programming evolution. And he taught us, the first machine we programmed, was an IBM 403 accounting machine. No, first he taught us how to use an IBM model 026 Card Punch, and then he taught us how to program or wire the boards on a 403 accounting machine, and when we mastered that he taught us how to wire a 407 accounting machine. And then he said, 'Now to show you that addition and subtraction are the basic elements of computation, here is, I'm going to have you do, here is a set of partial differential equations and here is the way we are going to solve them, and now I want you to wire a program on the 407 accounting machine to solve a set of partial differential equations.' And I can't imagine a better way to learn computing than from the signal pulses up. And that's the way Ken Iverson was, he was almost a physicist, I think his undergraduate degree was in physics, but he certainly had a physicist's view of nature. He dealt with nature from the atomic structure up.

Frana: He hasn't changed…

Patton: He was a remarkable guy.

Frana: Yes, you still have APL, Iverson's APL? Run it ?

Patton: I have it on my machine, yes, but as I said, APL, has this, the fault of its virtue, its ability to abstract is so incredible, it has about eighty commands, and of course, now knowledge engineering work has shown that even the most experienced human mind can only keep ten or twenty things as properly stacked, so even people that stress APL and push it will find that three weeks later, they can't read their own programs. It reminds me a little of the early days of word processing, always that program that use period and x for delete and so on, it was Micropro? Well, anyway, you remember one of those early word processors that had eighty commands?

Frana: Yes, that's right…

Patton: And no one could remember all those commands. I mean it was too many commands, and look now at Word 6, you have a two-tiered toolbar just to show you icons of all the commands and if it weren't for the mouse-over we couldn't figure those out…

Frana: That's right.

Patton: You have to mouse-over to see what that symbol meant.

Frana: Well, you've managed throughout your career to keep computing and humanities together in a way that's really extraordinary. What projects are you proudest of? We hear of your work with the PLATO courses, but over the years in that respect?

Patton: Well, I, when I was at Minnesota for a number of years I was not only director of Academic Computing, but I was a chair of The Center for Ancient Studies and this was a graduate research program, following Bill McDonald's idea. Bill recently passed away, but he was a professor of classics here at Minnesota and he wrote a book called *Progress Into the Past* and the thesis of that book is that by using modern scientific technology, which we normally employ to expand to our knowledge of the future, we can use the same technology to expand our knowledge into our past. He set up this program and recruited five graduate students and four of the five graduate students wanted to use either aerospace or computer technology in their research, and none of the twenty-three professors he had in the program knew anything about computers or aerospace, so he looked me up. I was a professor of Aerospace Engineering and Director of Academic Computing and I was a professor in the Computer Science Department. And so he asked me if I would join this group and I did. I was still active in doing research on Sanskrit and the Hebrew Bible. So, I joined and a few years later Bill retired and Professor Fred Luckerman succeeded him as director and I became Associate Director; and then Professor Luckerman became Dean of the College of Liberal Arts so I became Director. And I aggressively went out and sought grants for students and faculty. And we did a number of significant projects and the two I was most intimately involved with were a PLATO course to teach Sumerian cuneiform, and a PLATO course to teach Egyptian hieroglyphics.

Frana: And those courses have a legacy and they are still being used in a form. You remarked to me that there is a company, Learning Bytes is still using…?

Patton: Yes, Learning Bytes was a spin-off of Control Data. They bought the PLATO technology and developed it, into a rather than a course-based computer-based learning technology, into what they call Learning Bytes. That is, it's more of a training than an educational technology now, because these Bytes run nine to fifteen minutes and of course now the technology can be deployed on ordinary microcomputer networks, it doesn't need special PLATO technology. But two of their local clients are National Car Rental and Northwest Airlines. And the way these are often used is, if a counter agent at National or Northwest doesn't have anyone in line, there's nothing to do, the computer knows who is at this terminal, and they say you haven't qualified yourself on clerical discounts. Suppose someone shows up at your counter and he's a clergyman and he and his wife and four children, two of the children are under three, want to travel from Rochester, NY, to Sacramento. How would you put that? And of course, that's the worst case. I wrote an airline tariff thing one time and that's the hardest thing to compute. It will take a human four hours to compute that and we couldn't do it on the computer at all. And nobody actually does it they just guess, of course. So then she says hey that's interesting, go. And so it gives her about a nine minute expert analysis of how to book a flight for a family, how they do it, why the clergyman gets a ten percent discount, if his wife gets a ten percent, the children and so on. Or if it's at National Car, someone shows up, and they want to rent a van, to carry an eighteen passenger van, and they don't have one there but they know there is one in town. In fifteen minutes here is how they can learn to accept that order, get the van there on time, get the booking and have a charge for it. So it's just very, very practical. But I think

the way they are using it and the way we use PLATO at the University of Minnesota are vignettes, are little views into the future of distance learning. And I think one of the aspects of distance learning particularly in the industrial context, is its just-in-time nature. It's been shown many many times that people learn ten times faster on the job than they do in the classroom. And nobody knows quite for sure why. But I think the immediacy and pragma win over dogma in those situations. And the pragmatic nature of learning is something you are going to use the next time someone steps up to your counter. It has immediacy, and that, I think, recommends it.

Frana: So these products as they are used today, are these more simulation tools or are these actual expert systems?

Patton: Some are actually simulation tools; some have a nature of expert systems. I've been trying to introduce them at Lawson, because Lawson has the same problem every software company has. We get competitive advantage by pushing new technology in the business world. Lawson's the smallest of the Tier One business application software providers, but for twenty-six years we've always had the reputation as a technology leader. So we bring out this advanced technology in accounting, in customer relationship management, in activity based costing, activity-based management. And then because of that technical leadership in the application domain, we get good customers. For example we just sold twenty-two million dollars worth of software to Columbia Healthcare in New York. But a case that is more painful to me was our previous largest sale, a seven million dollar software license to Louisiana Pacific, the largest paper manufacturer in this country. So they bought our software because of advanced features and installed it. I was on an airplane a few months later and I was sitting next to a woman who was working away on a ThinkPad and I was reading something and

she noticed it was a Lawson document and she said, ‘You work for Lawson?’ I said, ‘Yes, I am the chief technologist.’ She says, ‘Well we use your software at Louisiana Pacific and it stinks.’ I said, ‘Really, well, what’s wrong with it?’ She says, ‘Well, I’m manager of environmental components and every time we build a new paper mill, and the milling paper involves lots of hazardous chemicals, processes,’ she says, ‘I have to not only meet all of these state and federal requirements for environmental compliance, but I have to come up with a precise cost, capital cost and run out cost, amortization cost of this technology over the life of this plan.’ And she says, ‘I know the information is in our databases from previous history, but I can’t get to it.’ And I said, ‘Why don’t you use our activity based costing technology?’ She said, ‘Well I heard you have that but we haven’t installed it.’ She said, ‘Well as a matter of fact, what our chief financial officer did, is install Lawson software right up to the point he had everything we had in the previous software and then he stopped because it was too complicated.’ I said, ‘Well look, I’ll send you a care package with documents and white papers and you can consider them arrows in your quiver.’ But you see the problem; the fault is partly his and partly ours. We need to have built in to our software these Learning Bytes which will allow a person like that, when no one’s looking, to take a quick look and see how does activity based Lawsonware look, see. You know**…**

Frana: Because otherwise she doesn’t know?

Patton: Yes.

Frana: And she won’t find out.

Patton: She won't find out.

Patton: You know every field has its traditional story. On the one end, accounting is very telling and it applies to this case actually, to that chief financial officer. The story is this: in this thirty person accounting group in this old-fashioned company, the chief accountant comes in every morning, unlocks his roll-top desk, and unlocks a little drawer at the top of the desk, opens the drawer and looks in and closes it and locks it and then the day's business begins. Finally, the old gent dies, and his Number Two in command, waits a respectful time, like about three days, and he goes and asks the widow if he could have the key that her late husband always carried in his left vest pocket. She says, well yes, I still have his clothes here; I'll go find it. She comes back a few minutes later with the key and hands it to him and he thanks her. He rushes back to the office, the whole staff gathers around him. They open the roll top desk, they open the little drawer, they pull out and inside there is a 3x5 card that says, "The debit side is near the windows". (Laughter)
And of course, if it's a sea captain it's, "Starboard is the right side of the ship."

Frana: Yes.

Patton: So you see what that guy had, he had Learning Byte, you see, and he looked at it every morning and it got him through the day. Well, this is what we need and we all have this problem with Microsoft. Microsoft is its own worst enemy. I finally have NT 4.0 running and I finally know how to use most of the stuff in Office 97. Because of my position, Microsoft sends me a free copy of Windows 2000 and a free copy of Office 2000. Have I installed them? No way! Why should I take any risks, you know, why should I open Pandora's box and take this risk? Now I am going to have to

do it eventually. People are beginning to send me attachments that are getting harder to open. So I'll have to do it eventually. But even technology heat seekers are reluctant to rush into this stuff if they don't have the immediate need for it. So what you've got to do is create the need. And I think education creates the need. You know Socrates said that the higher we climb the mountain the further away is the horizon. And I think we need to give people easy climbing lessons by these Learning Bytes. So I think Learning Bytes is a fascinating thing. I think you really ought to talk to Steve Shamblott and Rajiv Tandon, who are the co-CEO's of this company. Rajiv taught at St. Thomas over twenty or thirty years and he is a very articulate spokesman for this technology.

Frana: And he's still local?

Patton: They're both local. They're out in Hopkins or somewhere. But, I don't have their cards or anything but you can look them up in the phone book and I think they would make very good interviews for what happened to PLATO. Now, there have been some analyses about what was wrong with PLATO. PLATO is another one of these technologies that could only be deployed economically in large scale. And of course all technologies begin that way. In the Middle Ages, the only clocks were one in each village in a tower, right? And people in the field heard them strike. And now we carry clocks on our wrists. Although technologies always begin that way and they downscale. PLATO began as a mainframe technology; the break-even point in the first PLATO system to make it economically viable is four thousand terminals. And it took a huge Control Data computer to handle the four thousand terminals. At the University of Minnesota I talked Bill Norris into funding us on a project to make a freestanding PLATO terminal. We got some money and we got mostly parts. We got a Winchester disk, in those days a Winchester disk was five inches thick

*Revised 2014-01-10 to correct spelling of Steve Shamblott, p.14.

and twenty-four inches in diameter. CDC gave us a PLATO screen and keyboard and a box and we used the AMC 8000 chip, which executed Pascal P codes and built a free-standing PLATO system as a terminal. This was before the IBM PC's. Apple II's existed, but of course you couldn't do anything like this on an Apple II. So we migrated some PLATO courses to this freestanding terminal, and we brought up the Sumerian and Egyptian courses on this. And the Egyptian course is hard for PLATO, and it was hard on this. And then we had show and tell day, and so Bill Norris sent two senior vice-presidents and two vice-presidents and two or three technical directors out for the show and tell. So we showed them the stuff in the morning, what we had done with their money, what we had done with the technology. This was before the PC was introduced so that would have been late 70's, maybe 80. And then we took them to lunch at the Campus Club went back and we chatted for a while. Then our salesman, who was driving the van, went out, they all got in the van and they all went back to the Control Data tower and about five o'clock he called me. I said, 'Well how did it go?' He said, 'Well, we drove all the way back to the tower and no one said a word. It was absolute silence. And as we drove in the circle drive at the tower to let them off, one of them said, 'There's no way that's ever going to replace mainframes!' And the others all agreed.

Frana: [Laughter.]

Patton: Sad isn't it?

Frana: At the time…

Patton: Yeah. So you understand how a prophet can show someone something and they can't see it. They have an eye and see not. They have an ear and hear not. Right.

Frana: You mention you haven't installed Office 2000 or Windows…

Patton: Yes.

Frana: The office package yet. How do you feel about the Microsoft antitrust suit?

Patton: Oh…well…

Frana: I know you used to establish desktop standards at the University of Pennsylvania, so I would guess, …

Patton: Yes.

Frana: Does this complicate things at all?

Patton: Well, I …Microsoft has always been, from the beginning, they've been a predatory company. There is a great apocryphal story of Silicon Valley that illustrates this. Now this didn't actually happen. But the story is that Bill Gates and Steve Jobs, both college dropouts, got together to do a job for a small company and Bill Gates sold them this small embedded special purpose operating system for five thousand dollars. So he brings the job back and shows it to Jobs and so

Steve Jobs writes the embedded operating system, checks it out and Bill takes it back and he gets the check and deposits it and withdraws seven hundred dollars and hands it to Steve Jobs and says, 'Here's your half.' It kind of shows you the nature of these two men. It shows Jobs incredible technical brilliance and his naiveté, and Bill Gates mastery of sharp business practices. And I've met both men.

Frana : Oh, you have.

Patton: And, Steve Jobs is just amazing. They both are masters of PR; they both can really project a PR scene fantastically well. I did a demo three years ago, where IBM asked its four hundred partners of development to recommend demos for San Francisco Technology, which was a new distributed object frameworks technology they were developing in Rochester. I brought you some material on that.

Frana: Thank you.

Patton: This is for you.

Frana: Great.

Patton: And by the way I could, I could spend about maybe an hour sometime telling you about that and who you should talk to about that because this is the most important software development out of Minnesota.

Frana: This San Francisco Project?

Patton: Yes. And I was a member of the technical advisory board from the beginning, still am. And, I developed a whole new approach towards an accounting system and my assistant and I wrote a demo on it and we prepared the traditional two-minute elevator speech to show this at Comdex, at the IBM kiosk in the IBM booth. And so here I am, demonstrating this and His Billness comes up and he sees the San Francisco logo and says, what is this? And I introduced myself and explained it to him and I am halfway through a two minute elevator speech and he turns around and he sees a bunch of people playing a computer game by Tom Clancy called Perestroika and he just walks away right in the middle of the sentence and goes and sits down and plays the game for twenty minutes. So this guy clearly is 'a toy', you know. San Francisco is a greater threat, Distributed Object Frameworks, is a greater threat than anything he was doing and he couldn't stand still for two minutes to see it. Or maybe, I don't know, maybe he's much smarter than I am, obviously, maybe he just dismissed it after a minute. But, I think Bill; people think of Bill Gates as a technical genius, I think of him more as a business genius. He's in a way, the whole approach of his company has the same business plan that an Elizabethan boarding pirate. Sight Spanish ship, hail Spanish ship, board Spanish ship, take gold, kill crew, sink ship, look for next Spanish ship. I mean it's that simple isn't it? I mean he's done that again and again.

Frana: Now what, what has this meant though, for you? Microsoft, at least, that was a standard.

Patton: Well, I think Microsoft, as a standard is a good thing. And of course, that's why people tolerate this. That's why anyone tolerates a monopoly. I mean, in the early days of electric power, Boston, New York, and Philadelphia were 110 volts DC. Detroit was 110 volts, 25 cycles, and who could make radios? Who could make appliances? And so when we took the whole country to 110 volts, 60 cycles, that standard simplified building equipment. And that's kind of what Bill Gates did by gaining this monopolistic control. Now monopolistic control in and of itself is not a bad thing, but abusing that the way he did is a bad thing. And I think, I think the government and others are ganging up on him. And if you saw any of the trial and saw any of his testimony, I sat there watching it and I thought he just doesn't get it. Now, I was part of an effort to get thirty new instructions built into the Pentium VI to make Java execute faster. And Andy Groves testified at the trial about that and said Bill Gates called him and said he'd better not do it, so he cancelled the project. So, the one thing I personally was involved in came up within trial, I mean it was just **…** it was**…**

Frana: Very clear**…**

Patton: It was vintage Gates.

Frana: Very clearly an abuse.

Patton: Because Gates didn't want anything out in the marketplace that would make Java run better. And of course then Gates, he couldn't kill Java, he couldn't kill Scott McNealy, he couldn't kill Sun Microsystems, and so he decides to co-opt Java by bringing out Java with some custom changes to

it. Well the court said, well you can do Java, you signed a Java license, but either you do Java or you do something else. If you're going to do Java, this is Java. And so Bill Gates just picked up his marbles and went home and cancelled all the MS Java stuff.

Frana: It sounds like the next big threat is open-source software, I don't know if you saw the *New York Times* article on Monday, but the federal government now is recommending that its units be more open about sharing code and that there's a huge software gap and the private industry ought to get into this, more extensively than just a couple of applications. Can you comment on that? Does that sound like a good idea?

Patton: Well explain to me what you mean by open-source? You mean ah**…**

Frana: To share the code. And ah**…**

Patton: Like Linux, you mean**…**Linux?

Frana: Or like the Navigator browser, that's all open-source.

Patton: Oh yes. That's all open.

Frana: And to make money in other ways through the service…

Patton: Yes, yes.

Frana: And to leave the software be open. And, you know, given what you know about patents on software yourself. Does this strike you as the right thing to do?

Patton: Well, you know, there's a fascinating idea. Have you heard of Richard Stallman at MIT?

Frana: Yes, sure.

Patton: He's a MacArthur Fellow. He has several pieces of open software out. One is new which is a Unix-like operating system GNU, a recursive acronym that stands for Gnu is Not Unix. Of course, he has an editor, the best editor out, called Emacs, and when you get software from Stallman you have to sign a "copy left."

Frana: Okay.

Patton: …as opposed to a copyright. And a "copyleft" requires you to give it to anybody that asks for it and if you find any errors in it requires you to give them back to him so that he can fix it. Now I see a battle coming in this country on intellectual property law as it relates to software. And maybe intellectual property in some intrinsic sense**…** you know if you look back at human history you see that in the agrarian age the source of wealth was arable land. If I have 800 acres, I can grow corn on it, or I can grow soybeans on it or I can lease it to you, but I can't do all three. If in the industrial age the source of wealth is capital. I can buy 1000 shares of IBM, or I can buy 1000 shares of Microsoft or I can lend the money to you for some purpose, but I can't do all three. But the anti-intellectual

property group, the Stallman group says, wait a minute. Of course, the source of wealth in the information age is intellectual property, but if I have an idea I can give it to you and I still have it. And you still have it. What's the problem? I've worked on behalf of Medtronic on some twenty-one patent litigation issues involving pacemaker circuitry and we have that same cultural clash, that's happened in biomedical technology. You have the physicians who for five hundred years have invented new things and have named them after themselves, like the Miller forceps, okay. And then you have these electrical engineers who invent pacemakers and other technology and they patent them, okay, and they get a seventeen-year exclusive right a use of this thing, a monopoly. And this culture clash is happening in the courtroom. And, it's finally kind of working itself out. But I see the same kind of clash happening and if you explain to one of these intellectual property revisionists, let's call them, like Stallman, you say wait a minute. Look at the problem. Medtronic spends several years and five million dollars developing a new digital chip based pacemaker that senses bodily activity and does all these new things and it really, you know, is significantly better. And then they have to spend another five million and eighteen months going through FDA approval and so they've got all this money invested so they have to have a monopoly for a certain period of time to get that money back. And Stallman says, hey, just eliminate the bureaucratic crap, you know, that's what's costing us money. Well, Stallman is an accomplished anti-bureaucrat; obviously, he wanted to come up with a term like "copyleft". But it's not that simple.

Frana: How did we get to the point where Amazon can patent one-click software?

Patton: Well, this has been developing several years. I got a couple of very early software patents. And the guy I worked for at Medtronic who was Chief Patent Counsel was John Rooney and I had

met Rooney when he was a young lawyer and I was a young engineer at Univac. And, Rooney developed a very clever thing. When someone would come to him with this fantastic idea of doing something on a computer and they would show him a program**…** Rooney is kind of a Socratian guy, and by the way, he's also a Lutheran clergyman, as well as an attorney. And Rooney would say, 'Well, this is excellent, this is really nice technology, but could this also be done in microcode.' And the engineer would say, 'Well, yeah, yeah, here's the way you do it in microcode.' He says, 'Well anything that could be done in microcode could be implemented into static logic couldn't it?' 'Well, yeah here's the way you do it in logic.' He says, 'Fine, would you, would you write these up as three functionally identical but different embodiments of your idea?' And the engineer would say, 'Yes sir, I'll do that.' He'd come back with him and Rooney would file at disclosure on this hardware, which never existed and never will be built, with an alternative embodiment, in firmware, and in volume software. So Rooney was getting patents before the Patent Office knew they were giving them away. Well, but on this Corvet, now I've left these materials, most of this stuff in here is for Elizabeth is on Corvet and she may want to talk to me on this stuff to make sense of it, but **…** Some of these patents were on Corvet which was an automatic program generator. And I remember going back to the Patent Office, first of all we sent in the patent application and the patent examiner said, if I understand this it's not patentable. So I did a demo, actually using the actual system and I videotaped it and we took an application that was done by a jewelry manufacturing company in Hopkins. Every morning at eight o'clock the owner came in and called London and found the price of gold and then he entered that into a Univac computer and everything that we shipped after five o'clock that day that was made in his small jewelry factory the gold was priced at the price of gold that day. And this diligence on his part had built him up trust with his clientele, the jewelry stores. And so we took that application and we did it in Corvet and created a COBOL version and

videotaped a temp worker who'd never seen a computer before doing it. I took the videotape back to the Patent Office and they said bring a videotape and monitor because we don't have that there, we are just paperwork intensive here. I showed it to the examiner who had spent thirty years as a COBOL programmer, one of the first people doing software patents. He got so excited before the end of the videotape he said, 'Let's see the program!' He took the COBOL program and read it like a dime novel and turned around to his credenza and by then he was standing up like Neil Sedaka plays the piano you know, and he wrote an eighth claim for our patent. Since then another factor that has entered, along with software patents, we have, we have a mechanism, basic technology patents, on a mechanism, things like a transistor or so on or things made of transistors, or things made of pulleys, ropes, gears, levers, these basic mechanical or electrical elements. Then we have process patents. Here's how you make Blue Rain shampoo. This is the process, okay. And then we have method patents. Here's Karmarkar's method program for solving a linear program. Well, what's happening in the software, there's no confusion on mechanism. I have talked myself blue in the face trying to explain a CMOS circuit to a judge as a machine. And you can explain it to him as a sixty-four-layer photograph, a metallic photograph, but you can't explain it to him as a machine. But it is a machine, it really it is a machine. So the mechanism aspect of a computer program which John Rooney was taking advantage of for his patents, because that is all he could do in those days was patent a mechanism. That's not an issue. But we do have a lot of confusion. There's a twilight zone, a shadow zone, right now between these process patents and these method patents. And so we get silly things like Amazon.com patenting a one-click order and then getting a temporary injunction preventing Barnes and Noble from using it and it takes Barnes and Noble about three days to figure out how thin this patent is. So they bring out the website just like it looked before but instead of one-click order it says it's express order. So that's a pretty thin patent when you can just change one

word and escape, don't you think? But I tell you what's going on Philip, when that happened, every software company was terrified. And we're already beginning to get infringement letters at Lawson. And we just laugh about it. They're laughable. For example, we got one the other day from this company, from legal counsel for a company that had had a patent on data mining for activity-based cost data. And they said, it's come to our attention that Lawson Software does data warehousing and you have activity based costing. Therefore, we assume that you are infringing our patent. Well, so I just gave my opinion that activity based costing was invented by Taylor in 1910 and data warehousing you know, has been around for a long time and data mining, and all these things go much further back than a three year-old patent. And anyway, what we'd been doing, we'd been doing before their patent, so we're not subject to any infringement on the basis of it, but the whole software industry is in a tizzy on this and it is definitely a problem. And, but it's not a technical issue, it's not a legal issue, it's a cultural issue. Just like doctors and engineers arguing over intellectual property in the courtroom. And the problem in our legal system, in a precedent legal system like the English, Canadian, Scottish, American system, is that the body of precedent sometimes follows technology by twenty to fifty years. And it's taken at least thirty years for medical electronics to shake out. And, but the rate of change of technology is accelerating and courtroom delays are decelerating, so I don't know what the Omega point is going to be. When the Omega point is going to happen on it.

SIDE 2 :

tape 1:

Frana: Okay, well now I want to go back in time. In a lot of ways we've been talking about the present. We first started out; you set up a computer center for the University of Kansas.

Patton: Yes.

Frana: And it was an IBM 650 that was installed there. Could you tell us a little bit about that machine? And there's this impression out there that it was a very personal computer. A very large computer, but there's something in the hearts of many that …

Patton: The 650 was the first computer I worked on that had a quality that we call today "user-friendly." It was very easy to use. And it wasn't designed as a computer per-se; it was designed as a calculator to be used with a room full of IBM tab equipment. But rather than being electromechanical, it was electronic. But it had cards in, cards out, and it had a plug board that you wired. But in addition, it had this, these two thousand words of magnetic drum memory. It was called an IBM 650 Magnetic Drum Data Processing Machine. It wasn't even called a computer. And as I mentioned in some of my notes, it was designed by Engineering Research Associates in St. Paul, on contract to IBM. And then, Univac was purchased by Remington Rand and they ran out of money making the Univac I and then Univac bought Engineering Research Associates and the day after that was announced a man from IBM showed up here in St. Paul and handed them a check for two hundred fifty thousand dollars for the 650. Arnie Cohen tells that story. He's a trustee, I think, of the

CBI, you may have already talked with him, but Arnie tells that story with great gusto. And Arnie is one of the three names; Arnie's is one of the three names on the patent of the magnetic drum for computing. But then later, of course, Univac brought out a machine called the Solid State 80 and the Solid State 90, which use the same basic kind of architecture. But the 650 was a decimal machine. Now this was one of the major factors of its user friendliness because you didn't have to convert something to octal or binary to be on the machine, you put in decimal numbers and the machine computed in decimal numbers and you used a special code, called the biquinary code which is, of course, the same code that the Chinese abacus uses. And it was wasteful in terms of bits, and like all other user-friendly features it was costly to implement. It took more hardware but it sure saved on the man-ware. And the 650 didn't have tapes, it didn't have complex peripherals, you just put cards in got cards out, you could plug these boards but the average 650 programmer didn't, as a matter of fact the average 650 programmer programmed the machine in an interpretive language like SOAL, SPUR or IT internal translator and later a FORTRANSIT, which was an adaptation of the then very new IBM 704 FORTRAN compiler, FORmula TRANslator, which instead of translating to machine language translated to internal translator or IT. Internal Translator was an interpreting system for the 650 and but this was transparent to the user. You put in a FORTRAN program you got out an internal translator deck. You preface that deck with internal translator, which was an interpreting system. You put in an Internal Translator copy before your intermediate language deck, after your intermediate language deck you put your data, a ten inch stack of cards back in the card hopper of the machine. And, after a few minutes you start getting output punch from the other end of the machine. And you took that output over it put on a 407 accounting machine and printed your answers. This was, compared to using an IBM 701 or 704 or 705 or Burroughs Datatron or Harvard Mark I or even a Univac I, this was so simple. See Univac I, was much easier to program because it

had an alpha-numeric programming system. For example, later assembly language routines actually mimicked the machine language of the Univac I. For instance, the 'add' of Univac I would be coded A and the A was interpreted as an 'add', to multiply an M, to divide a D, and so of course this made it easy to program a Univac I, but if you wanted, once you wrote your Univac I program, you had to go back to your machine which converted your program onto a magnetic tape. Then you took the tape to the machine and then the output came in the form of another tape and then you had to take that to another machine to print the tape. And you were carrying around all these tapes and people in those days were just much more accepting of and used to punch cards. As a matter of fact for the first twenty years I was in computing, around my right wrist I always had a half a dozen rubber bands and I still sometimes have the feel of those rubber bands around my wrist. You carry them on your right wrist because you had a deck of punch cards in your right hand you took your left hand put a rubber band over them and pop. But now I haven't seen a punch card in ten years. (Laughter)

Frana: I think there are still a few out there.

Patton: There are a few, but…

Frana: A lot of library books still have them. And people were especially fond of what they saw, the interface, and the biquinary…

Patton: Yes, the binary quinary presentation on the 650. The 650 ran a lot slower than a 701 or 705 or a Univac I and so as it functioned you could see the light patterns on the maintenance console, the operator maintenance console, and repetitive operations were very, very noticeable. I mentioned to

you that I wrote, I helped write up an interpretive system at Boeing called SPUR. And one of the, the longest routine in this was the arctangent, that was the hardest thing to compute and I thought it was kind of interesting because it took eight minutes and nineteen seconds to compute about the same time as the Toccata & Fugue in D Minor. Because it was an iterative computation it was fugal in nature, so if someone would be walking through the machine room into the programmer room, and see it going he or she would rush to the door and shout, "It's on! It's on!" and, oh, fifty or sixty programmers would rush out and gather around the console of the 650 and ooh and ahh as the lights blinked and we saw the arctangent compute. Now some other routines had similar patterns because they didn't go long enough, you know they weren't recognized. But nonetheless, the blinking lights were fascinating. On even IBM's first machine, the 701, it ran so fast that while it had these blinking lights on the console, they went on and off so fast you couldn't see them. If you stepped through a routine looking for something you could see the lights change. But if you just said go, the console won't show it because the pulses were getting there too fast to light the lights, they just didn't last long enough until you saw any red neon bulb lit up. But on the 650 you saw every bit go by.

Frana: That's a great story.

Patton: Yeah, yeah, but I think the user friendliness of it was, it was so easy to use. And for a university, for the University of Kansas it was just a great machine to introduce people to computing because you could teach them a wee class in programming and they could go down and write programs and they'd run and they'd get results and it was, compared to the rest of computing it was kind of like slow motion, it was like doing something in a slow motion movie but it was still incredibly useful.

Frana: And then you installed a CDC 6600 at the University of Stuttgart while you were there.

Patton: Yes, right.

Frana: How long have you known Seymour Cray?

Patton: I, at that point, when I came to Univac in 1961 as a principal programmer, Seymour Cray and Bill Norris and those guys, Frank Mullaney, all those guys, had left five years before to form Control Data, and their first machine was the 3600 and so on. And so I didn't know Seymour Cray at the time, but I knew of him, because I was responsible as principal programmer for the programming aspects of the fleet test which was done with these computers onboard ship that had been designed by Seymour Cray. And this machine was the AUSQ-17 and interrupts were then known. There were machines that had interrupts, but in the late 50's Seymour Cray wouldn't use them. And so here was the world's largest real time system, a command and control, tactical combat system, which did everything with monitored buffering, no interrupts. And we used to joke in the NTDS system the tightest loop in the whole system is up refreshing the CRT screen, you know. Forget the torpedoes, you know; first we've got to refresh the screen. (Laughter) And so it was from inside out, you know, it grew like an onion, and only the outer layer defended the system. I was involved in the efforts to redo this whole thing, the one thing we learned on the fleet test, we passed the fleet test and the NTDS system became, we got an order for it, but we substituted a new computer, the AUSQ-20, later known as the Univac 1206, which had an interrupt structure and it had a different quinary factor but it, that machine was much easier to program the system. My role in that was checking out the

CS-1 compiler, CS-1 was the compiler that was built for the Navy. And you know in the early days of computing this has been lost, I think, this situation, but every big project, it seems like to me, every big project I was involved in this happened. You would define a problem, you would design a computing solution to this problem, you'd be given a set of hardware in which you were going to solve this problem, then the programmers would all sit around and design a new programming language in which to solve this problem. And this just happened again and again and again. And if you look back at the history of programmer-oriented language development you just see this proliferation of languages.

Frana: Which Jean Sammet made a career out of it, documenting.

Patton: Yes. Yes, documenting all those languages. And I think I told you the story about I took a class in computer design from Howard Aiken, the last class he taught at Harvard. And it was really a thrill to take a class from this guy but in a way I was amazed at what an interesting combination of advanced technologist and Luddite this man was. And I later, during the 50th anniversary of the Eniac, I had the opportunity to work with J. Presper Eckert quite closely. And I had known J. Presper Eckert since my days at Univac and worked with Eckert before, and always admired his incredible intellect and memory. But I decided in many forms getting ready for the 50th anniversary of the Eniac then two years hence, I was having dinner with him and I said, you know you told me that the first problem that was ever solved with the Eniac was Edward Teller and [Nicholas] Metropolis came to Penn at the Moore School and they wanted to do a simulation of the H - Bomb on the computer. They had to learn to program it themselves because there was no way to get security clearance for all the programmers. And the programmers, by the way, were Hester Eckhart and his

first wife, Adele Goldstein and Kay Mauchly. You know, the wives of the three guys who did the ENIAC were the world's first programmers. But they couldn't get clearances for them; they couldn't get these Cosmc clearances or Q clearances it took, in the ADC days. So Teller and Metropolis had to be taught how to plug wire a program into this machine. And I said 'Pres, did you, when that happened, did you have the idea of solving the problem on the computer?' Because the Eniac was built to create firing tables for artillery. I said, 'I studied two years with Howard Aiken and I am convinced he thought that the purpose of computers was to create tables of functions which you then used along with Whittaker and Watson in the old British separation of variables technology for partial differential equations, to come up with series solutions and then you have a table of Spherical Bessel functions, spherical Hankel functions, Legender polynominals and you plug the numbers into your tables and you get the solution at every point in space. I am convinced that Howard Aiken, in spite of his genius, never had the idea of solving the problem on the computer.' And Eckert said, 'You're right Peter, he didn't, and we did, and we couldn't explain it to him.' You couldn't solve a problem on a computer, a computer built tables, but he said Teller and Metropolis didn't know anything about computers, and they didn't know any better, so they solved the problem on the computer. And they and John von Neumann went to the AEC and the Congress and they got money to build computers. They convinced the government that this new technology was worth something.

Frana: Even though Aiken**…?**

Patton: Aiken, I remember one time in class someone asked Aiken about what he thought about Grace Hopper and her new compiler languages, Math-Matic and Flow-Matic, and he just got livid. He was a very very tall thin man and I remember the purple ring around his mouth, he was just

*Revised 2014-01-10 to correct spelling of Kay Mauchly, p.32.

bright red except for this purple ring. And he says, 'Don't you ever mention that word or her name again in this classroom!' But it was just anathema, the idea of a complier**.** Programs had to be programmed in machine languages, you see. The idea of simplifying the language was just anathema to him. I saw him blow up two other times when someone referred to "data storage" as "memory." He says, 'Young man I'll not have you using anthropomorphic terms in this classroom.' And then the other time someone referred to an Eccles-Jordan trigger-pair as a flip-flop and he got hysterical. I don't see why that was so anthropomorphic because Aiken himself described the function of an Eccles-Jordan trigger pair as a married couple in bed and they mix up the controls to their electric blankets and she gets cold so she turns hers up, and then he gets too warm so he turns his down, and then immediately they rush to an extreme. Now if that isn't anthropomorphic, (laughter) bedroom analogies, I don't know what is. And yet, we couldn't use anything like that.

Frana: Apparently as long as he came up with it, it was okay.

Patton: Yes, but he was a brilliant guy. What he made us do, he divided us into groups, and each person designed a four-bit module of a computer. I was in the group on four-bit adder. This was a chassis which we punched with Greenleaf punches and put eight octal sockets in it and install 6V6's and wired up these things and then we had Jones' terminal strip from the back and then the whole class got an A or a B depending on how well this worked when we wired it all together and it did a four-bit computation. So in 1971 when someone rushed in and showed me all excited the Intel 4004, which was a four-bit integrated circuit computer, he said, "Have you ever seen anything like this?" I said, 'Been there, done that.'

*Revised 2014-01-10 to correct spelling of Eccles-Jordan, p.33.

Frana: Created, made one.

Patton: Yes, I made one of those.

Patton: I was at the ’71 Comcon meeting in San Francisco. I was giving a paper on socially, on a floating-point arithmetic in a associative memory parallel processor for the Goodyear Staran machine. For radar correlation in a missile defense system. And Robert Noyce gave the keynote. And he had just announced the 4004, the first real integrated computer that did something useful. He gave this fascinating talk and he got only polite applause. From the back of the room there were 3000 people in the room, someone said, “Mr. Noyce, this is very impressive but what in the world is something like this going to be useful for?” And Noyce just held it in his arm and his coat sleeve came down and you could see his fist and his wristwatch. Dead silence. Nobody got it. Then he pointed to the wristwatch. Nobody got. He said the first application of this kind of integrated [circuit] will be to electronic wristwatches. The volume in that [circuit] will drive the cost down where we can then begin to use them in everything else. Three thousand people laughed at him.

Frana: No way.

Patton: Isn’t that sad? I mean if there had been a mud pit handy they probably would have put him in upside down like the Israelites did Jeremiah for telling them what was going to happen if they made a treaty with the Egyptians. I mean, I was, I sat there flabbergasted. Here was this genius and three thousand of the top circuit designers in the world laughing at him because they didn’t get. Well, they later got it. He was right.

Frana: Speaking on your work on missile defense, could you talk a little bit about Project Safeguard? That's something that I don't know very much about at all.

Patton: Well, Safeguard was a very interesting project. I got involved in it when I was still at AIC and after I came to the university I couldn't shake it. I kept getting called back to Huntsville, Alabama, by Clarence Giese who was the civilian head of that and there was a Colonel, I can't remember his name at the moment. He kept getting me to do these studies. The basic problem with Safeguard is you had up in Grand Forks, North Dakota, you had a thing that looked almost as big as an Egyptian pyramid. And on the north, east and west sides it had huge circles on which were radar antennas. And this thing was a flat-top pyramid, like a Meso-American pyramid but it had slanted sides like an Egyptian pyramid. And inside the pyramid was a control computer, and the control computer was a Control Data 7700 which was a one-of-a kind machine, consisting of two Control Data 7600's - the fastest machines in the world at the time. And a large core memory. And this radar, this $ 236 million dollar radar built by General Electric could do eight thousand beam switches a minute. And the Control Data computer, one of the 7600's, could actually program and schedule 3500 beam switches so here you had the most expensive and largest electrical appliance in the world, which would sit there searching for incoming missiles and then not know what to do next and so it just, it had an electromechanical sub-program built into it, that when the computer died it just shot a beam of power right up into the air. And then the computer started scheduling it again and then we started searching and controlling and so on. So this was the worst case of system unbalance I've ever seen. So the task I was working on and this documentation I've given you describes some of those projects, was to figure out how to employ the massive parallel resources of the 7600. The

7600 had two adders, two multipliers, and one divider, of course, but no one knew how to invoke all of the arithmetic capability on one machine to this problem let alone how to put the problem between two machines. Because in some sense scheduling is an intrinsically sequential activity, you know. If we want to schedule something we have this cloud of potential events, like we are going to have a commencement at a high school and so there's all these things that have to happen, so there's this cloud of all these events that have to happen, but when we schedule them we map them onto the time axis, this happens, and then this happens, and then this happens and then this happens. That's an intrinsically sequential process. Now at big conferences we have multi- we call it parallel tracking, where we just have multiple sequential tracks. And so what I did was I took the algorithm, I studied the algorithm, I found out what concurrency was available in the algorithm then I tried to map this onto the parallel resources of several machines. Of course, we tried to make the 7600 work, then we tried to create one, we tried the Texas Instruments Advanced Scientific Computer. We worked with the GE Staran. There were a couple of other special purpose machines; I don't know the names of them. But the idea was what can we do by matching algorithms and hardware to be able to make good use of this system**.** And I think that the design of the system was good, I think the way it was partitioned was good, the ability to**…**you see once you started detecting**…** the problem is in the beginning when you have an empty sky and you get information that a missile or missiles took off somewhere in the Soviet Union you began looking for incoming things. There is no partitioning there. You have the whole northern sky, I mean, from the horizon to 45 degrees, and you are looking for something up there. Now as soon as you see something it probably becomes partitionable, so it becomes practical, soluble. Then you can see in the things here we were able to solve it. And here's the solution: First of all you start tracking all these incoming missiles and so you are out of search mode and not track mode. Okay, so you are not scheduling unknowable or potential events you are

watching something that's falling. So you know where it was and where it's going to be. And so the radar scheduling is much simpler than that. And it's a piece of the pie. We're now like in the commencement analogy, we now have multiple parallel sequential affect. Now then, you start taking the data you're getting from these objects you are tracking and you discover that some are decoys. So then you stop tracking the decoys, just forget about them, and you start putting more resources on the ones that aren't decoys, the ones that actually have weight because they are probable warheads. Okay they start getting closer and now we are down to half a dozen things we're tracking. And what the heck we have eight thousand beam switches a minute; there's plenty of resource. Then we go to the hard part of the problem, the hard part of the problem was the Nike X system which sent up small missile interceptors to get close enough to those atomic warheads to explode the small atomic warhead and blow up the big one. Now, in my opinion, the mathematical probability is a little weaker and it took a lot of competing resources to do that. But by then you knew you were going to try to blow up six things and you had half of the trajectory in which to do it, and you have all these missile batteries sitting around, so then the thing was busy controlling particular missiles and shooting them off and trying to get intercepts. And you didn't have to hit the thing; it's not like today's Smart Rocket technology. It's not even like the Scud, these things are going off in outer space, we aren't talking about nuclear pollution here. I mean you blow this thing up you know 150 miles out of the atmosphere, I mean, so you, I mean you blow up a nuclear warhead and it blows up another. So I think that the system was workable, but I think, let's face it, Safeguard was an electro-political problem.

Frana: Yes, it's cancelled, I discovered, the day after it's declared fully operational. Now how is that?

Patton: It was okay, but it was politically incorrect. And look at it today, isn't it the same thing? We have an electro-political problem. Now here I think we have a double fault, I think the electrical thing, the electrical part is dubious, and the political investigation is even more dubious. And I think Reagan knew that and he was probably suckered in. I think he was just saber rattling, but he had a loud saber, and the people knew there was one in there.

Frana: Quite a bluff.

Patton: Yeah, it was a great bluff and it really worked. And I think it was a factor, it was a major factor in the demise of the Soviet Union. But now we have some new players and these players, now I don't understand, I'm no politician; I did not understand why the Chinese and the Koreans are so incredibly intimidated that we might have a perfect defense against their weapons. What's the matter with that? But they are, and I mean, they'll just come right out and say it. We are upsetting the balance of power on nuclear defense. So I guess the whole thing is a game of cops and robbers. You know.

Frana: Speaking of cops and robbers, this is a terrible segue I know, people today are very interested in computer security and this is something you've been dealing with for a very long time, and figuring out ways for police to collect information and yet provide security over that information is something that you've been involved in down here. Can you comment on it? Sort of where we've been and where we are now and whether it matters.

Patton: Yes, well I promised to tell you this anecdote.

Frana: Yes, I want to hear it.

Patton: Okay, and I have here the materials on HEJIS which is the Hennepin Justice Information System, which I designed, about the last thing I did at Analysts International. The Hennepin County Police Chiefs Association hired us to do this Justice Information System. And I tell you police chiefs are really security nuts. That's what they do with their lives. So they kept wiring more and more and more security into my information system. And finally it got to the point where it was unusable. So I got up in a meeting with these twenty-some police chiefs and I told them what they'd done and in order for them to understand it I constructed the following metaphor: I said, 'Can you imagine a library. Now libraries have tremendous loss of books. People steal books, they forget to bring them back, and they have rare books and they keep the rare books in special rooms and you have to get permission to go in these rooms. There are various levels of security in the library. But can you imagine a library that is so secure that you can walk into the lobby and through glass, bullet proof glass windows, you can see the books, you can see the card catalogue, you can see the stacks, but you can't even get in and even look at the card catalogue. Now that library is secure, it's not going to have any losses. You say yeah, but isn't it useless? Aha. I said that's what we've built here. We've built a system that's useless. Your policemen can't do anything with this.' And they said, 'Well but we have these security problems and so on.' And so I said, 'I wasn't involved in the solution of this problem, but I did some interviews and some work for the law enforcement assistance administration LEAA, which was funding this whole thing.' I said I was out in Los Angeles and the police chief

there told me a fascinating story. He said we have an LEAA grant and we have these miniature teletype machines in our police cars. The biggest risk we have here, aside from domestic violence situations, if a policeman is likely to get killed it's mostly in domestic violence situations, but the second most likely thing is to make a routine traffic stop and walk up to the vehicle and ask for a driver's license and get shot in the chest. So, in order to prevent that they wanted to be able to pull up behind a car, key in a license tab and then cross reference that immediately to an auto license/ownership file, cross reference that immediately to a driver's license file and then check if there are any outstanding warrants. That tells the officer whether he should approach with extreme caution or with his gun dropped. They were the test pilot for that system and they had become extremely embarrassed because if these policemen saw a good looking woman driving a Cadillac convertible they'd pull her over, check her license tab, check her driver's license and discover that she had several arrests for prostitution and then get out and approach her car and give her a choice, I'm going to throw the book at you for all these things or you can drive with me to the nearest motel for forty-five minutes. Well they were getting away with it until they stopped a very attractive young woman driving a Cadillac convertible who was a former prostitute who was now married to a municipal judge. You can imagine how that police chief felt about that. I tell you that's the worst case, in their world. And the solution, I volunteered the solution before he told me it, I said isn't it obvious, the solution is logging. Log every transaction and tell the officers you are logging it and that's all you need to do 99 times out of 100. If anyone doesn't get the idea, you've got convicting evidence. And if you can't put him in jail, you can certainly fire him from the force. He said you're absolutely right, that's what we're going to do. So in many of these sophisticated information systems you are handling restricted data, the arrest records weren't even confidential. That was public data but the problem in this information age, making public data instantly accessible is a

difference of degree that amounts to a difference of kind. And let me give you an example. Several cases have gone to appellate court in Florida. Seven years after a person has served a conviction for a crime, that crime is taken off their record. So if you'd go down to the courthouse and look for it you couldn't find it. But if you contact the National Crime Information Center at the FBI, it's there, because federal law supersedes state law and there's no federal law that has any seven-year amnesty right. Well another situation, and this ended up in federal court. An aerospace engineer in LA got a traffic ticket on mistaken identity and protested it and the officer considered it giving him cheek, so he arrested him and took him to jail and the next day the man was freed by a judge and the officer was reprimanded. The victim asked the judge, since if this arrest was legal if he needed to report it. He said no, in California we have a process by which we expunge arrest records for unjustified arrests and he said we'll do this process right now and they went with the process and it was all duly witnessed at the time. But unfortunately the night before, that arrest had been recorded with the National Crime Information Center of the FBI in Washington. Well, you know an aerospace engineer is just a high paid migrant worker, and of course when this guy's project was done at Lockheed or wherever he was, he went and applied for a job at Northrop and he had a letter signed by that judge that said when he filled out an employment application he could check that he'd never been arrested, because under California law he hadn't. So when he filled out the application, he checked he'd never been arrested and Northrop refused to hire him because he had been arrested, because they did an NCIC check for arrests. And so he sued the State of California, Northrup Aircraft, and the FBI and that case was in court for years. So that's an example of how you can destroy someone with something that's true and a matter of fact, law makes a difference between libel and slander. Libel is something that is not true and destroys you, slander can be true and it destroys you. And from the oldest Accadian Law, from the law of Hammnurah: slander, using true

information to hurt another person has been a punishable offence. Of course in Accadian law you had your tongue cut out, even though he told the truth but you did it to maliciously hurt someone.

Frana: Do you know how that was settled?

Patton: I don't know. I didn't follow that case and I don't know what the legal precedent is on that now, but I do think that**...** You know we introduced this with a talk about cops and robbers and the whole computer security thing now is strictly a game of cops and robbers. Every time they close the holes some smart hacker finds a new one. At Lawson Software every week we upgrade our McAfee virus killer system to a new level every so often, I think we are up to 7.29 now. And every week we upgrade with new virus protection our system library of viruses and I think we are up to 7029. So it's just a constant effort to keep ahead of these viruses and so on.

Frana: Were you involved in University security while you were here at the University of Minnesota?

Patton: Not at the U, it wasn't so much an issue here, because everything was pretty much hard wired to each terminal, but at Penn, we were early users of a fiber optic cable system that had 18,000 IP addresses on campus. We were early involved in that, and we got into big trouble because we had to make it easy for people at the university. We had what network security people call an open back door. Our front door was ok, we had a front firewall, but we had an opening in the back door. And what happened was a professor of computer science in the Netherlands got through our back door to get to Stanford and he put five "bombs" on the Stanford's administrative system, which was the

same one we had at Penn, using a large 3090, and then he wrote a letter to the President of Stanford University telling him what the logic bombs were and when they were going to go off, offering to remove them for five thousand dollars each on or before the day of the explosion. Well, this first came to my attention when two FBI men and the Commissioner of Police at the University of Pennsylvania and the top-networking director at Stanford showed up at my office. Naturally I did everything they said and so we found out where the back door was and closed it because we were about to be thrown off the Internet because of this. Then they went to Interpol and once we closed this and the guy no longer had access, they were able to go in with "electronic bomb sniffing" tools and find them at five places and eliminate them at Stanford. Once they had the evidence and they had the letters and everything they went through Interpol and went to the Dutch police and the Dutch police told them very briskly that what this man has done is not a crime in the Netherlands. That's essentially why the 'Love Bug' guy got off in the Philippines, what he had done at that time was not a felony in the Philippines. It is now. It wasn't then. Now I think now his career is probably badly tarnished, but still he caused billions of dollars in damage and he's Scot-free.

Frana: Yes he is.

Patton: So you see, this is another extension of my comment about the immediacy of information. Now we have the ubiquity of that. You can destroy people halfway around the world you don't know, with information that may or may not be true. You get really down to the basic fact that in addition to technology we need stability and I think there's a lot of stability in this approach.

Frana: Otherwise there's no solution other than cops and robbers.

Patton: Yeah, other than cops and robbers.

Frana: Busting people.

Frana : Well, we've got a little time left, you've reviewed the course of your career. Are there any generalizations that you haven't remarked upon so far, especially as an administrator in education, leaving aside the Lawson years and all, the majority of your career was in that role. Forgive me if I am giving you a term that you don't like.

Patton: Yes, I would say that I've spent much of my career in Development. My main interest was in applications, and I was really interested in applying computational technology to the educational process and educational administration. I think the real breakthrough in this came in '69, when Bell Lab scientists, Dr. Pierce, published a report called the Pierce Report; he used an example of an ideal computer service, The Dartmouth Model, which is a very good model, where computing was made truly available to all the students that are in Dartmouth University, just like the library. This came to be known as the Pierce Model or the Library Model of Computing. Well when I was hired at the University in '71, my mission was to build a library model for computing at Minnesota. And that's the challenge I took. Well at the time, computing for undergraduates, you see, doing this at Dartmouth which was a small primarily undergraduate institution was one thing, here at Minnesota which has a huge undergraduate population it's quite another. Now the legislature had the attitude in 1971 that they would fully fund undergraduate instructional computing and their full funding in '71 was 800,000 dollars a year. But they would not spend one penny on research. And they interpreted

graduate, even master's degree thesis-related papers, as research computing. At the same time the federal government had been supporting computing for graduate research primarily through the Atomic Energy Commission. The old AEC and the DOD, and that was changing. And the AEC, even in '71, was already expecting a new Department of Energy that was saying, well, we'll fund this ten cents on the dollar. Well the legislature wouldn't put a penny on it, not ninety cents on the dollar; they wouldn't put a penny on it. So, computing at the University in '71 was almost completely dominated by physics and chemistry graduate instruction and chemical engineering, but there was no component outside that. So I decided the first thing I had to do was, before we could have universal funding, library-like funding, in computing, we had to have a library-like access on demand. But the library, you know the library, was a radical new thing four hundred years ago. (laughter) But you know it's kind of built itself in. Although education is very, very conservative, I mean, after all they still call what we do in the classroom a lecture, which is a Latin word for *reading the book* to the students, and now each student has the book you don't have to read it to them, we still call what we do a lecture. But, so my first step, was to build a foundation with the House of Higher Education Appropriations Committee, that master's degree research computing was an educational expense and therefore fully fundable. And over the next two or three years I was able to double the legislative component by 800,000 to 1.6 million.

Then, I worked with professors in other fields, and having been a humanities computing person myself, I just couldn't make any progress. I'd go to these people and show them what the possibilities were and they just couldn't see it. So I finally decided, you know, I got the idea from Robert Gordis. Robert Gordis wrote this wonderful commentary on the Book of Job. He's retired now but he was a professor of Hebrew at Union Theological Seminary. And he wrote two books on the Book of Job. A commentary about two inches thick and a popular book called *The Book of Man*

and God. In his commentary he writes, his dedication is: “To my parents from whom I learned much, to my teachers from whom I learned more, but mainly to my students from whom I learned the most.” And I thought this is how professors learn, they learn from their graduate students. So, I started hiring graduate students at the University Computing Center. Instead of just hiring from physics and chemistry, I hired from other fields. I hired Sara Graffunder, an English graduate student. She retired as Director of Systems Programming at Cray Research. She’s now running The Science Museum of St. Paul. I hired Vicki Walsh, from Ancient Greek Archaeology. So I started hiring these students from humanities areas, and then when I had the chance to join the Center for Information Studies that was where I made my big move. Then I started working closely with graduate students and I saw these graduate students convert sixty-five and seventy year-old professors to computing, it was just amazing, just amazing. I thank Robert Gordis for this. So then we had universality and we had demand. But the problem then was, it was very hard to get money. It was getting increasingly hard to get grants and contract money for computing. And other universities were offering fifty cents on the dollar to granting agencies like The National Endowment for the Humanities, the NSF, and so on and the University of Minnesota was offering zero, nothing, one hundred percent pay, so I thought how am I going to get around this? Well, there was only one solution to that problem, and that is to gain the competitive advantage. So I decided to gain competitive advantage by buying a used Cray -1. We bid on the Cray-1, it was ten times more cost-effective than the Cyber 74 and it was fifty times more cost-effective than a crop of VAXes that most universities were getting. And so NSF decided they’d rather fund something here, and pay full price on the Cray, and buy some department in Wisconsin a VAX. They could get more for their money. And so that really made the difference. I couldn’t change the funding algorithm. So what I did at the end of my thirteen-year tenure in that job, the budget had gone from 800 thousand a year

*Revised 2014-01-10 to correct spelling of Sara Graffunder, p.46.

to 13 million a year. The state component had doubled and the rest we earned. So, and I was at a meeting, I was on the Lax committee which set up the NSF supercomputing program, because I was running the only supercomputer in a university in the country, so they invited me here. I was obviously the granddaddy of this. But they didn't like my attitude on this. Jim Infante, who was head of Math and Science Division and some of these other people they said, 'Well we just can't give researchers money.' I said, 'Wait a minute. Explain this to me. You give this guy five hundred thousand dollars to buy this lab equipment, to do this experiment, and hire these graduate students, why can't you give him twenty five thousand dollars to buy computer time?' 'Well he might use it irresponsibly. He might strike a deal with his provost, the provost says, well you can use that money to hire more graduate students I'll give you twenty five thousand dollars worth of computer time.' I said, 'What do you care? You get value, what does it matter?' Oh no, we're going to pass out the time; we have these huge committees of fifty people that will allocate times so there will be no stewardship issues with this. So I made my impassioned plea for the free market economy on this thing and one of these physicists from the East Coast, Professor Patton, he said, "I think you must be an ancestor of Adam Smith." I said, "Don't you mean a descendant?" And he blushed bright red and he says, "No, I mean, in your case, I mean an ancestor." And I said, "Well I am Scottish, like Adam Smith and I must confess I agree with every word he wrote." (laughter) And so they did it their way and I think it was a disaster. I feel that same way about checking travel expenses, you know. Here you pay someone $100,000 a year. You trust him or her to bring in 10 million dollars worth of business and you fuss with them over a three hundred dollar travel expenditure. Give me a break, I mean, that's a, you know it's just a poor focus of priorities, you know.

Frana: Do you feel like your experience was emulated elsewhere successfully?

Tape 2

SIDE #1 (#3) :

Patton : Oh yes. I sold one computer to the University of Stuttgart and in the contract, Univac wrote that if they had any troubles in running it, that I would go there and run the system for them. This was such an embarrassment to IBM because IBM's plant in Europe was in Sindelfingen which was the same suburb. No it was an enjoining suburb to the university's Aerospace Division, which was in Vaihingen. And of course having right in the back yard, having this huge Univac computer. And the further embarrassment that it was a showplace for Univac for all of Europe, so visitors were coming everyday to see this thing. And IBM was doubly embarrassed because they had offered the University of Stuttgart a free 7090 and instead the university bought a Univac 1107. So finally IBM a couple of three years called in everyone that worked in the Computing Center at the University of Stuttgart and offered them a job at exactly twice the salary they were getting at the university. And so the university invoked the contract and I was moved from London where I was running sales and technical support for the Univac British Commonwealth, for a large scale systems, to Stuttgart. So my wife and four kids picked up and moved to Stuttgart. And I took over the center and I immediately hired four Norwegians, an Egyptian and an Israeli, who wouldn't talk or look at each other, until one day because of some misaddressed mail discovered that they had a common Yemenite great grandmother. I thought that was a wonderful ecumenical small world story. But anyway, I developed that center there and then in '66, I left and came back here. I came to the University of Minnesota in '71 as the Director of Academic Computing and built it up and got the Cray-1 in '81, and the University of Stuttgart got a Cray -1 in '82. And then we got serial 003 Cray-2, then the University of Stuttgart got serial 007

Cray-2. At the 25th anniversary of the Computing Center in Stuttgart I was invited back to give the keynote. And I felt like there is no greater satisfaction in life than building a program, an organization which outlives me, and the fact was still emulating what I was doing. I mean they were copying every move I made, you know, from 3000 miles away. So that, that was kind of rewarding. And I think in the days of large-scale mainframe computing the University of Minnesota had the reputation as being the premier computing institution. And yet we had less academic and legislative funding than any other major Big Ten operation. At that time Henry Koffler was Provost and I can remember every meeting I had with Henry Koffler went like this. He was a man that wanted to be in the middle of the pack, and if you'd go in and tell him that we needed a new so and so. You had to be ready because his question would be, 'Where do we stand in the Big Ten on this?' If we were better than number five forget it. If we were number five, he'd think about it. If we were number eight, he'd give you a couple of hundred thousand dollars to come up with a plan. I mean he was a stickler and worked on competitive advantage. And then Peter Roll and Carl Adams, and I did a study for Nils Hasselmo, when he was Vice-President of Planning before he went to Arizona to become provost for Henry Koffler when he went down there as President. In this study, I've given this book already to the CBI, but we did this study on computing in higher education and we came up with four responses. This is a time when Dartmouth was pushing ahead, and the University of Houston was doing things, and Drexel was talking about issuing a Mac to every student. We came up with a plan, for how the university could compete. And the first thing we did is factor the problem into four responses: The minimal response to this technology challenge was reactionary. That is, we will engage in some new technology only when we are dragged into it kicking and screaming by legislative, organizational, competitive requirements, legal requirements, only if we have to. That was the approach that Penn took about eliminating social security numbers of students

*Revised 2014-01-10 to correct spelling of Nils Hasselmo, p.49.

as ID numbers. When it became clear it was illegal, it was against the 1936 Social Security Act, then they had to stop doing it. They figured out something else. But until then it was done. Then the second was competitive. That was Henry Koffler's mode. Do we have to do this to maintain competitive advantage in our field? Forget the rest of the world. If we're in the Big Ten, are we number five or better? If we're number five or better then okay. Then the next was intensive. And that is, do we launch in faith? Do we invest in a technology because we think it will give us a competitive advantage, not just maintain, but give us additional competitive advantage to reduce costs or increase customer, or in this case, student satisfaction and enhance educational effectiveness and so on. This is a Robert Solow approach. Using technological substitution for material, capital or labor. Robert Solow won a '85 Nobel prize for that idea, and I use that idea. Then there is what Carnegie Mellon, and the University of Houston, and MIT with Project Athena, and Drexel with the Macintosh Program. And that is pioneering not just launching out in faith, but really pushing the envelope. Spending millions. MIT spent 100 million dollars developing Athena. Now can you imagine any president of the Big Ten spending a hundred million dollars on computers? No way! Now if they got 50 of it from IBM and 50 of it from DEC like MIT that doesn't matter. The source isn't even the issue. That's, that's pretty amazing. So, where do we want to be? And what we recommended to Vice President Hasselmo, that we should be intensive in our computing approach.

Frana: Thank you Peter.

Written Replies

Attached below are written answers to CBI interview questions submitted on August 24, 2000.

Your first major at Harvard was Sanskrit. How did you come to the decision to change over to Engineering and Applied Physics?

I studied Sanskrit as a means to learning Navya-Nayya Logic, developed by the Indians as an extension of Aristotle's logic based on his 25 syllogisms. The Indians learned Aristotelian Logic from the Greek scholars that Alexander the Great took to India with him. Indian Vedic scholars used their new logic to develop the Upanishads, comparable to the Jewish development of the Talmud as a unique-logic-derived commentary on the Torah (except of course, the Talmud anticipates the First Order Prepositional Calculus developed many centuries later in the West). The Sanskrit professor at Harvard was a leading authority on this ancient logic system and agreed to teach it to me if I majored in Sanskrit. One evening when leaving the Sanskrit library, I walked by the Harvard Computation Laboratory and through the window saw a priest working on the Harvard Mark IV computer. The contrast of this (then) modern machine being used by a man wearing the garb of a magistrate from ancient Rome was too much for my curiosity so I went in and asked him what he was doing. He very graciously explained that he had encoded the Hebrew text of **The Book of Job** as two decimal digits per character and was doing a computerized linguistic analysis to separate the tri-literal Semitic stem of each word into Hebrew, Arabic, Chaldee, and Elamite. I thought that at last here was an intellectual prosthesis that could help me with Sanskrit, so the next term I took a programming course from Ken

Iverson, then a graduate student. Over the next summer and Fall term I switched my major from Sanskrit to Engineering and Applied Physics and took all computing courses except for a couple of major required courses that I could not test out of. The Dean of Engineering said that in the whole history of Harvard nobody had ever changed his major from Sanskrit to Engineering, and while he made a bit of a fuss over the whole thing he was clearly fascinated by the way I dealt with the bureaucratic obstacles.

You have been active for a very long time in the study of ancient languages. What is the relationship between your interests in computing and in lexicography? In computing and the humanities? Why was SNOBOL so useful to studies of languages? Could you briefly describe the PLATO instructional system? What has the legacy of PLATO been?

The course project that Iverson had us do was to create a KWIC concordance of **Faster than Thought** by Bowden on the Univac I. Nelson's had just published (1956) a concordance of the Revised Standard Version of the Bible that had been prepared on a UNIVAC I in 1000 hours. General Groves, CEO of UNIVAC, had just given Harvard serial six UNIVAC I. Later I went on to work on a concordance to **Finnegan's Wake** by James Joyce. I learned SNOBOL, but like most early programmer oriented languages it was limited and clumsy, so I did everything in machine language. I never actually used it for a project. The Plato system was the most sophisticated computer based educational system ever developed and the technology is still being used by LearningByte International for industrial training at NWA and National Car Rental, among others. I suggest that you interview Dr. Rajiv Tandon, CEO at LearningByte on this topic. I got some funding and developed a course in Sumerian Cuneiform which had a character

based dictionary built in. We were trying to find the limits to Plato technology for CDC but we didn't even come near the edge in this project. I sold CDC a second project to develop a course to teach students to read Middle Kingdom Egyptian Hieroglyphics. This proved a bit tougher but Ken Decker, a graduate student in Ancient Studies developed an Egyptian Hieroglyphic font designer and editor for Plato which was excellent. It was so good that Brill Publishers in Leiden, Holland licensed it and our character set to publish books on Egyptian language and literature. Both courses were used successfully at the University. The dictionary for the Sumerian course was a real technical challenge for me so I continued to do work with Miguel Civil at the Oriental Institute in Chicago and with Ake Sjoberg at the University of Pennsylvania on Sumerian Lexicography, until 1996, in fact. My students and I published several articles on Sumerian Lexicography, especially on Emesal, the women's language.

You seem to have received a broad undergraduate and graduate engineering education in the 1950s and 1960s. Could you speak to that?

I went to Harvard just after James Bryant Conant introduced the new general education curriculum. Harvard had reformatted the old state university style BSEE, BSME, BSCE, BSChE, BSAE, etc. five-year majors to develop a generic Engineering and Applied Physics four-year major. It was wonderful, and the teachers were terrific. They taught us engineering as a set of general principles and the mathematical background to support them. As an engineering intern in 1955 I designed my first and only vacuum tube logic circuit, since then I have had to learn transistor circuit design, TTL, ECL, NMOS, and CMOS. I was able to do so easily because I was taught the basic physics and mathematics of electronic circuits, not just some cookbook recipes.

The curriculum well prepared me to do an MA in Math, an ABD in Math and finally a Dr.-Ing. in Aerospace Engineering.

I've read through your class notes for Ken Iverson's Applied Mathematics class at Harvard. Did he ever talk about APL while you were there?

Ken never used that term in his course but years later when I read **A Programming Language** the approach was very familiar. He was developing that technology from his early graduate research and teaching days at Harvard. He later went on to become an IBM Fellow. I was an avid APL user at one time and still have it on my desktop machine. My only complaint with APL is that it has the fault of its virtue; its power of abstraction is so great that one cannot read one's own programs after a lapse of only three weeks. The most amazing APL program I ever saw was written by Prof. Delaney in Sociology at UMN. It was 12,000 lines long and modeled body language in the job interview process. As director of UCC then, I had to buy him an APlum compiler from NYU to compile it at Minnesota on the Cyber 74 after he was recruited.

You organized the first computer center at the University of Kansas which relied on an IBM 650. I'm told that the 650 required optimum programming and insertion of instructions and data into specific memory locations in order take advantage of drum rotation. How was this done? How did the 650 compare to the UNIVAC Solid State computer? Or was that available yet? Did just the appearance of biquinary blinking lights really help market the machine?

Early IBM 650 programmers coded referring to a drum latency card and an 8x10 sheet of paper with 2000 numbered rectangles on it as a map of the drum surface. For example if the multiplicand was in drum location 432 and the multiplier in 621, then the product could not be stored until at least 39 drum timing slots had gone by, so 661 or any location number in that row on the drum surface map was the optimum place to put it. Bell Labs developed an interpretive system called SOAP that did all of this plus floating point arithmetic and functions for you but it took up 1200 of the 2000 locations. At Boeing Wichita we set (and met) an arbitrary goal in 1957 of replacing the function of SOAP with a system called SPUR that did all of these things in only 600 locations. I think I gave my SPUR manual to CBI.

The UNIVAC Solid State 80 and 90 were much later but were the same concept as the IBM 650, because they were designed by the same people at ERA. Arnie Cohen can give you the full background on the 650 and SS80, his name is on the patent for the magnetic drum used in these machines. Fred Lang and Rich Daly developed the first Enroute system for the FAA using the SS80, their approach with the printed out paper strips is still used in the control tower when the radars fail! (An SS80 no longer prints them out, however.)

There are a few people who remember the 650 as akin to a "personal computer" because of its accessibility. At a half million dollars apiece that's clearly misguided. Why do they remember the 650 this way do you suppose? What was it about the machine that was remarkably different?

The machine was the cheapest on the market, in the same class with the Burroughs Datatron 200. It was designed by ERA here in St. Paul under contract to IBM with a contract that allowed IBM to give ERA a percentage of the profits or just pay them a flat fee of $250,000. The day after UNIVAC bought ERA, a representative of IBM appeared at ERA in St. Paul with a check for $250,000. IBM vastly underestimated the sales of the machine, assuming they would sell only 200 or so but they sold 1500. The machine was different because it was in the same class as all the other IBM electro-mechanical tabulator equipment except that it was punched card in – punched card out machine and was an electronic computer. They later offered it with a FORTRANSIT compiler which translated FORTRAN to IT (for Internal Translator) which was an interpretive system similar to SOAP or SPUR. This made it even easier to use so it had a sort of user friendliness like today's desktop computers.

The blinking lights on the IBM 650 were truly spectacular. In SPUR, the longest routine was the floating point arctangent function, which took 8 minutes and 19 seconds to run, the same length as Bach's Toccata and Fugue in D minor. Naturally the light display was fugual in nature since it was an iterative routine. Whenever anyone happened to notice the arctangent pattern on the console they would burst into the programmer's area and shout "It's on!" We would all run into the machine room and stand around the console going "Ooh" and "Aah" just like we were watching a fireworks display or listening to an organ concert. I think that was the most beautiful program I ever wrote.

What was the nature of your work in developing computer aerospace simulations in the late 1950s, particularly for Boeing and at the Midwest Research Institute in Kansas City? What is the Aronzajn-Patton Method for computer based flutter analysis?

I did the dynamical simulation for the wet wing B-52 G and H which store fuel in fuel cells in their wings to gain an around the world flight duration capability. This led me to an interest in the aircraft flutter problem. Boeing seconded me to Kansas University to set up their IBM 650 center, the second 650 in Kansas. There I met Nachman Aronazajn a noted authority on variational methods in Hilbert Spaces. He needed someone who could actually show his methods worked on real-world problems and I had real-world problem that could not be solved. I did an MA in Math with Nach but he couldn't support the extension of my method for non-selfadjoint operators in Hilbert Spaces as a purely mathematical undertaking. I later found a mentor in John Argyris professor of aircraft structures at Stuttgart and did an Dr.-Ing. with him. John kindly christened my method the Aronszajn-Patton Algorithm. It was the first computer method able to do flutter analysis on an aircraft with more than 20 degrees of freedom. At MRI I worked of the reconstruction and analysis of two Super-G Constellation crashes for TWA. Grisly but important work for an aerospace engineer,

The CDC 6600 did not employ paging or segmentation even though these techniques were well-known. Why was that?

Seymour Cray was always reluctant to use new technology, especially if he didn't invent it. He didn't use interrupts on the AN/USQ-17 for the Naval Tactical Data Systems, and as soon as he left Univac to start up CDC we redesigned the machine to become the USQ-20 or Univac 1206 which had interrupts. When he designed the Cray I, his first semiconductor memory machine, he didn't use parity bits dissing criticism by his now famous comment "Parity is for farmers." Serial number two Cray I had parity, and Seymour admitted his mistake.

In 1961 and 1962 you wrote compilers for the Naval Tactical Data System. What were some of the special programming needs the DOD had for collecting and processing combat data rapidly? What is deghosting?

I went to Univac as Principal Programmer in 1961 with the mission of checking out the Fleet Test System using the AN/USQ-17 and the CS-1 compiler. This system test was complicated by the fact that NTDS was entirely in octal notation and arithmetic, by the fact that the code sequence for CS-1 did not distinguish between 0 and O (same six bit code), and of course the lack of interrupts on the computers. Sorting targets with alphanumeric labels was a problem (with 0's and O's) but I solved it by some of the semantic language processing techniques I had developed for literary analysis. Programming technology in those days was focused on new language development rather than understanding the relationship between algorithms and data structures. Witness the development of CS-1, CS-2, Algol, Jovial, Fortran, Cobol, etc.
CS-1 was a lot like Bell Labs "C" in that it generated about three ML object instructions per source statement rather than six like Fortran and Cobol did.

You installed a CDC 6600 system at the University of Stuttgart in the mid-1960s. Some of the designers of the 6600 have argued that its Chippewa Operating System benefited from Seymour Cray's singular vision. The same people blame System/360's design-by-committee management structure for poor system performance and inability to do time-sharing. Is that a fair assessment?

The Chippewa Operating System was highly tuned to the processor architecture of the 6600 but when you had to do disk or tape I/O it just rolled over and played dead. In fact the elegant mini-computer based I/O system of the 6600 was developed by Jim Thornton. I gave my copy of his book to the CBI.

It is not really fair to say that the 360 was designed by a committee. The ***goal*** of this architecture was to subsume as a single successor to the decimal 7070 series, the character 7080 series and the binary 7090 series machines that IBM made. Naturally the architecture was eclectic by its very nature, and its ancillary new language PL-1 was an interesting programming language development. PL-1 was intended to succeed Fortran, Cobol, and Algol and had an eclectic mix of the features of those three languages.

Gordon Bell has said that by being larger than all of the other computers of the day the 6600 pushed the limits of reliability. How did it compare to the reliability of other machines, notably IBM's Stretch or the later CDC 7600?

Seymour took a non-conventional approach here. He told me that he knew he could not get feasible MTBF numbers at a reasonable cost with the technology he had to use to get the speed, so he elected to go for an unprecedented MTTR value of 15 minutes. The combination of 12 hour MTBF and 15 minute MTTR was a technical triumph. The 7600 used this same design approach, and although the 6600 in its maturity did much better than 12 hours MTBF, the 7600 never did significantly better than that. The Stretch was very complex in its look-ahead buffer, and other architectural innovations but it had a lot of error detection circuitry in it. Unfortunately this made it too expensive to produce and IBM had to stop making it because they lost $2M on each system they "sold." I gave my book on the Stretch development project to the CBI.

What was so revolutionary about having the random access memory hierarchy under program control in the 6600?

The 6600 was the first true RISC architecture. At IBM in Rochester they jokingly define RISC as "really it's Seymour's computer." I did a test on the Univac 1108 when I was chief architect at Univac in Roseville in 1968 and showed that a Fortran compiler could generate only 25% of the instructions in its CISC ISA, but that a Fortran compiler could reach 100% of the RISC ISA of the 6600. Of course the secret of the 6600s speed was its interlaced memory design.

1966 you wrote a humorous column comparing American and European programmers for *Datamation* under the pseudonym Libellator. In that article you described European programmers as fiercely independent loners and Americans as team players. Has this generalization held up since then?

It is still largely true, however European programmers are a bit more team oriented that they were then. In Stuttgart in 1966 if you went into a colleagues office unexpectedly he would quickly turn over all the papers on his desk and greet you warmly. French programmers were very artistic and independent.

In the early 1970s you were director of the Dicomed Corporation here in Minneapolis. What kinds of computer graphics applications were early Dicomed scanners/image recorders intended for? What sorts of file formats and processing techniques were considered and used at Dicomed?

I got the idea from looking at some of the first NASA Mariner photos from Mars. The before pix looked like blobs and the processed pix were very sharp. My oldest son had just broken his leg in a sledding accident and I spent an evening with this crying ten year old and three hand-wringing physicians trying to read an ambiguous x-ray. The next day I told Fred Lang and the graphics team we were recruiting from Univac that medical image processing might be a good starting product. Unfortunately, while the radiology dept is a profit center in every hospital in the country, radiologists were very suspicious of this new technology at first because it allowed any physician to read an x-ray. The company just about went under until we sold some units to the CIA and the NSA. Our terminals did the image processing they needed (you could see an enemy tank under camouflage) but not having CRTs, they did not re-radiate information that could be scanned by an enemy.

Between 1972 and 1976 you consulted for the Army's Project Safeguard on operating system design. What was Project Safeguard and what was the nature of your work? Wasn't Project Safeguard shut down by Congress a day after it was declared fully operational in 1975?

The safeguard problem was simply that you had a $236M radar as big as a pyramid sitting around waiting for a $12M CDC 7700 computer (two 7600s connected by a large core secondary store) to schedule it. The radar could do 8000 beam switches a minute and a single 7600 processor could only schedule 3500 of those. They could not figure out how to engage the second processor, however the algorithm made good use of the two adders, and two multipliers on the 7600. I was hired to come up with a concurrent algorithm and test it on the serial one (prototype) Cray I in Chippewa Falls and the ASC (Advanced Scientific Computer) at TI in Austin, Texas. I had not realized that the CBI had any interest in this but I will bring my materials on Safeguard to you meeting and donate them.

Most of the first Cray-1s were used for defense and atmospheric research. What were some of the early tasks assigned to UM's Cray-1B in 1981?

I got the go ahead to get the Cray I to replace the Cyber 74 from the Budget Executive as a VAX killer. At that time the NSF (Gordon Bell) was pushing the DEC VAX on Universities by giving 50% funding for a 50% match to any department that requested one. UMN was trying to retrench $50M out of its budget and was very embarrassed by having to fork up $150K to Chemistry, Physics, Astronomy, etc for these machines which would further add to departmental costs to

operate them. They called me front and center and asked me for a creative alternative. I offered to borrow $5.5M from the St. Joseph Leasing Company and buy the Cray I on which I could then offer computing service ten times faster that the Cyber 74 and at a cost performance 50 times better than a VAX. It worked, they got no more departmental matching fund requests for VAXes and UMN became the first university to operate a supercomputer. Most of the workload was shifted from the Cyber 74 over two years and we returned the 10X cost-performance advantage to the users. Although educational computing was legislatively fully funded, research computing at UMN was not and had to be paid for at 100 cents on the dollar. This left our faculty at a competitive disadvantage in getting grants and contracts since there was no university "match," the Cray converted this situation into a competitive advantage.

What is the "Cray effect"? Is there a relationship between the Cray effect and the development of solutions for the automatic extraction of parallelism?

Highly parallel computers can offer concurrency in two dimensions; space or time. Parallel processors do so in space by replicating identical computational resources, e.g., microprocessors bussed together. The Cray offered multiple computational resources in time, by a process called pipelining or vector computing. The optimal vector length on the Cray I was 64 words or elements per vector. The algorithm designer's challenge was to "unroll" the computation in modules of 64 identical operations (e.g., multiply/adds) and pipeline these through the Cray arithmetic unit. The Cray's highly interleaved memory further supported this process by largely eliminating the memory/processor speed unbalance. It turns out that for most (sequential)

algorithms it is easier to extend or parallelize them in time than in space. Unfortunately supercomputers are made of high cost - low volume components and parallel processors are made of low cost - high volume components, so the economics of this situation are opposite the ease of use considerations.

For a long time it was said that sequential computing would always outperform parallel processing. Where did this perception come from? Did this have anything to do with the perceived inadequacy of parallel algorithms and languages?

Early computing technology was intrinsically parallel because algorithms were made to be executed by two hundred people (called *computors,* like *auditors*) all sitting at desks in the same room with calculators. As each mathematician or computor finished a calculation she passed it to the person at the desk to her right and accepted new "input" from the person on her left. Usually they were women mathematicians, that's why the first programmers (e.g., Adele Goldstein, Hester Eckert and Kay Mauchly), were women. In an early paper John von Neumann said that parallel computers would be much better but that the circuit technology of the day would support only sequential ones. The first computers had a single register, the Accumulator/MQ that often doubled as a program decoding register as well. My first few years as a programmer were largely involved in ***sequentializing*** these concurrent algorithms!

The term "parallel algorithm" is a non-sequitor because an algorithm is defined as a "***sequence*** of simple operations that can be executed by an automation to reliably produce the correct result every time it is used." We can understand the term a a poetic metaphor of what we would like to

have to guide the programmer's design of concurrent computations on a parallel processor. There is no such thing as a parallel program;

Concurrent programs run on parallel processors.

Amdahl's Law was first expressed by Willis Ware who showed that computing is a rate problem, i.e., if you can do some portion of your computation concurrently at rate R1 and the rest must be done sequentially at rate R2, then the overall rate R will be found from

$$1/R = 1/R1 + 1/R2.$$

For example, if we take a 200 mile trip in an automobile and go the first 100 miles at 100 mph, and because we throw a rod at that speed, finish the trip by driving all night at 10 mph to reach our destination, our average speed will be only 18mph. Ware showed that the sequential portion of the task always dominates the concurrent portion. This fact caused Gene Amdahl and other architects to put their effort into faster and faster sequential computers. Gordon Bell and Steve Wolfe at CMU developed the Cmmp parallel architecture which took 20 years to get from the lab to the marketplace. Interestingly Gordon tried three times as VP of R&D at DEC to bring out a parallel VAX but could not overcome the DEC culture's obsession with VMS. I bought beta versions of all three but was never delivered one due to this mistaken notion that VMS could not schedule concurrent tasks. They never understood that random scheduling of concurrent tasks on a parallel processor is ***within a factor of two*** of optimum scheduling (Leonard Cohn's theorem). Oh well, you can always tell the pioneers by the arrows in their backs.

Consider my domino theory of parallel processing first proposed at MCC in 1985. Suppose we had out on the test floor the perfect parallel processor. Now suppose that we had a four volume set of "parallel algorithms" analogous to the ACM four volume set of collected algorithms. The problem is now how do we communicate the concurrency in the algorithms to the parallel execution capability of the parallel processor? Once past this point we can depend on an operating system to schedule within a factor of two of optimum by Cohn's theorem, and of course the ultimate performance problem of PP is a packaging one.

Physical reality operates concurrently. As the wag said, time is all that keeps *everything* from happening at once. But our ability to mathematically model physical reality is strained through three hundred years of sequences and series, 100 years of algorithmic thinking and forty-five years of sequential Fortran programming. Can we expect to find much concurrency left in a Fortran DO loop? Surprisingly there is a lot, but it is far easier to harvest for a Cray-type vector architecture than it is for a spatially parallel architecture. The focus of my research program at MCC was on new languages able to express computational concurrence in a natural way.

A *Star-Tribune* article announcing your departure from the UM Supercomputing Institute in 1987 mentioned that you were engaged in an important survey the future of the American supercomputer market. What particular problems did the supercomputing industry face at that time?

Our legislative mandate required that the SCI support the Minnesota supercomputer industry so I went to Lloyd Thorndyke and John Rollwagen, CEOs of ETA and Cray and asked them what we could do for them. They both noted that 90% of the programs running on supercomputers were retreaded 7600, cum 6600, cum 7094, cum 7090, cum 709, cum 704 programs. Most of these programs were simply run on supercomputers for speed reasons but had never been modified to take advantage of the major architectural gain of the supercomputer, as opposed to the minor circuit technology gain. They wanted me to find out what would be the applications of their machines in the future as opposed to these 90% legacy applications. I started up this information service and in one year had sold $990k of subscriptions. The university became nervous about this unrelated business income and spun us off after we paid off their $50k investment. Within a few months we sold the business to Dun and Bradstreet's Dataquest Division and were retained as consultants. I went on from there to consult for NASA/JPL in Pasadena on supercomputing and parallel processing.

Computer security and privacy issues are very important problems today. Yet you have been talking about developing policies for collecting, accessing, and disseminating database information since at least 1971. When did computer scientists begin seeing data collection as a potential problem? Was it a response to popular pressure? What safeguards were initially tried?

My first brush with this problem was HEJIS the Hennepin (County) Justice Information System I designed for the Hennepin County Police Chiefs Assn. I gave all my HEJIS stuff to CBI. At each design review the Chief's kept adding security requirements, until the database became basically

useless. I compared the system to a library at which you could enter the lobby, but not check out books, go into the stacks or even look into the card catalogue. I showed them that data had to be managed and gave them some examples from police work. It turned out that most of the problems they had anticipated could be handled by logging. Most of what CJIS systems work with is public rather that private data, but publishing even public data on the Internet gives it a distribution that it never had in days of paper based courthouse records. Such systems had led to a number of federal cases entered by people who felt their civil and economic rights have been compromised by the accessibility and immediacy of these systems. We can discuss further if you are interested. I do have some amusing stories on this topic, i.e., gossipy cops and data security.

You hold two patents for the automatic generation of business software applications. The road to software patenting, however, has been a slow and tortuous one. How hard is it to prove your case today? What are your thoughts about how the software patent controversy played out over your career, and especially since the early 1980s?

Software patenting has come to a difficult crux illustrated by Amazon.com asserting their patent for "one-click ordering" against Barnes and Noble. This event has the whole software industry in a tizzy right now. The patent was so weak that it took BN only three days to figure out that all they had to do was change the button to say "express ordering" to avoid infringing the patent. The critical issue is that there are mechanism patents, process patents and business methodology patents. Many software patents seem to be a confused mixture of the latter two. Let's discuss more.

Given your interest in establishing desktop standards at the University of Pennsylvania in the early 1990s, do you feel that Thomas Penfield Jackson's findings of fact in the Microsoft antitrust case is a step in the right direction?

I agree with Jackson's decision. I have a number of interesting stories about the misuse of monopolistic power by Microsoft.

Reviewing the course of your career, are there some broad generalizations you can make about the changing (or not so changing) role of the information administrator in industry and education?

The role is changing from a technology based one to a mission based one. Not that a mastery of the technology is not still essential but the main focus now is on application of IT to the competitive advantage of the business or the institution. Today's CIO must understand the business he or she is in and be able to map technology advances into business advantages.