Oral History of Robert (Bob) Freiburghouse

Interviewed by:
Gardner Hendrie

Recorded: February 28, 2011
Chapel Hill, North Carolina

CHM Reference number: X6064.2011

Gardner Hendrie: Who's graciously agreed to do an oral history for the Computer History Museum. Thank you very much, Bob.

Robert (Bob) Freiburghouse: Thank you for asking.

Hendrie: All right. What I'd like to do to start with is get some of your very early history, maybe even starting with your family background. Where you were born and brought up, a little bit about any siblings you had. What your parents did, that sort of information.

Freiburghouse: Okay. Well, I was born in a small farming community in southern Wisconsin. My father was a town doctor, one of two or three, small community of 2000 people. I grew up, went to high school there. I have one sister who's two years younger than I am. Graduated, went to the University of Wisconsin for my freshman year.

Hendrie: Can I interrupt for a second?

Freiburghouse: Um-hum.

Hendrie: Did your mother do anything, or was she fundamentally a stay at home mom?

Freiburghouse: She worked in my father's office. She was a dual major. She had an English and Music major in college. But she worked in my father's office as a secretary nurse assistant.

Hendrie: So she had a college education?

Freiburghouse: Yes, she did.

Hendrie: Both your parents...

Freiburghouse: Yes, they did.

Hendrie: ...were educated. When you were in grade school or high school, I don't know when this would be, but what is your earliest memory of what you thought you would like to do when you grow up?

Freiburghouse: Well, that's interesting. Because I always was very much interested in political activities. The Cold War was the focus of everybody at that time in the '50s. So my goal was to work for the state department or the CIA. So I was a major history and political science buff since I was in grade school.

Hendrie: So you were always interested in history and politics...

Freiburghouse: That's what I thought I was going to do.

Hendrie: Those were the courses that you really liked in school?

Freiburghouse: Eventually, that's what I got a major in. It was not in computer science. Computer science didn't exist back then.

Hendrie: That's certainly true. What year did you graduate from high school?

Freiburghouse: 1959.

Hendrie: Yes, certainly, it did not exist then. So did you enjoy math and science? Or were you really...

Freiburghouse: I was afraid of math because I did poorly in math in high school. In retrospect, I think it's because I had a really poor math teacher. Because I never had trouble with math afterwards. But it steered me towards the soft sciences, as opposed to hard sciences.

Hendrie: Do you have any idea how your interest in history and politics and what was going on in the world and the history of it, lead up to where you were? Were one of your parents interested or—

Freiburghouse: No. I don't think so. I mean, they certainly were politically aware. But no. It was just something that I seemed to gravitate towards. Because it was in the papers all the time. There was a constant perceived danger during the early years of the Cold War. We performed nuclear attack drills and I volunteered for the Civil Air Patrol. We went up and looked for Russian bombers flying over the town. The whole country was worked up about it, as I was as a youth.

Hendrie: Were there any particular teachers in high school that you remember? That you really, sort of, looked up to or were really good that influenced you, possibly?

Freiburghouse: No. Although the social studies and history teacher was the one that was teaching the subjects that I most enjoyed. I had a fascinating little old lady who was my Latin teacher. And Latin was required when I went to high school.

Hendrie: Where did you end up going to college?

Freiburghouse: Well, as I said, I went to the University of Wisconsin in Madison for one year. I did well. But I wanted to get away from my small town. I just went into the Air Force recruiting office and signed up.

Hendrie: Oh, wow, active after one year?

Freiburghouse: Yes.

Hendrie: What did they decide they wanted to do with you?

Freiburghouse: Well...

Hendrie: I don't think there are any history jobs in the Air Force?

Freiburghouse: No, there aren't. I wanted to fly. I had glasses that were a half inch thick. At that time, there were some aircraft that had enlisted people as crewmen. But you had to be an officer if you were going to be a pilot. But there were very few planes even then that had crew. They were rapidly being replaced by electronics. But, anyway, that's what I thought I would do. Of course, you do what they ask you to do. So when I finished boot camp, and they gave me my assignment, nobody knew what it was. They give you this five-digit job number that describes what your job is. Well, it turns out that the job that they had given me was a punch card machine operator. So I was sent to a base in North Dakota that had a huge room full of IBM [International Business Machines] punch card equipment. All of the different types of machines, the sorters. The application that they were doing there was, basically, an inventory control application. Because this was a base that supported a wing of B52 bombers and a wing of fighter interceptors. They had to keep track of all the parts and things that were needed to supply the base. So that was the application. They didn't have any computers. They were scheduled to get one. I worked there for a month or two. I realized the procedure that they were doing with the punch cards could be greatly simplified. So I talked to the sergeant in charge of this thing and said, "Look, if we change this, this way, we can save a lot of time in running this whole procedure." He looked at me like I was from Mars. But he was willing to try it. And, sure enough, it worked. So I was an instant hero.

Hendrie: Very good. Just a new recruit.

Freiburghouse: Yes, a guy with one stripe on his sleeve. In any case, then, they were picking people to go to programming school. They needed people, because they were going to get this computer. So they gave everybody this programming aptitude test, which I failed. But the captain who gave the test realized that I had already shown real aptitude...

Hendrie: Figured something out. <Inaudible>...

Freiburghouse: I did something pretty difficult. So he said I could go anyway. Those of us that were selected went to a base near Kansas City to an IBM school to learn how to program the IBM 305 RAMAC

[Random Access Method of Accounting and Control]. I was the first in the class. So the aptitude test did not mean much.

Hendrie: In spite of the test, it turned out that you could do this quite well.

Freiburghouse: Yes, I did better than the captain, who also went. In any case, that machine was a vacuum tube based computer. All the circuitry was implemented with vacuum tubes. The main memory was a magnetic drum that had 100 tracks of data. It was a character oriented computer. It had, I think, 100 characters on each track of this drum. It had 100 pieces of core memory that were used as a buffer. So as data came from one of the peripheral devices, it would come to core. Then, from the core, it would be placed on the drum, as the drum rotated to proper position. But what was most interesting about the 305 was it was, I believe, the very first computer that used disks as the secondary storage. Each disk was a platter, I think, about 18 or 24 inches in diameter...

Hendrie: We have an early prototype of that at the museum. Yes.

Freiburghouse: Well, this unit that we had, I believe, had two stacks. One in the main cabinet and one in an auxiliary cabinet. I'm not sure, but I think it was 5 megabytes of storage per stack.

Hendrie: That's correct. You're right. I think I remember that number, too.

Freiburghouse: Anyway, the machine had these huge doors, with the vacuum tube circuitry in them. We used to leave the doors open to help cool the computer. The computer also had a plug board as part of the main program. So that each instruction had a single character field that would cause a signal to be emitted from the correspondingly labeled position on the top of the board. Then, you could wire that into various other functions on the board. So what an instruction did was determine by its op code, its addresses and by the signal emitted from this position into the wiring on the board. The computer came with a huge inventory control application already on it that had been developed by somebody, IBM or computer sciences or some government contractor. But our job was, simply, to run this thing and, of course, fix it if something wasn't quite right. And something wasn't quite right, because it would lose data. There were records on the disk that had gotten lost. We realized this, over time. So I came up with a program and a plan to recover the lost data.

Hendrie: Where was the lost data?

Freiburghouse: Well, the records were just not connected anymore. They were there...

Hendrie: They're still there?

Freiburghouse: Well, we had...

Hendrie: <Inaudible>...

Freiburghouse: We had to assume they were still there. But they were just lost and that they weren't connected properly.

Hendrie: Okay.

Freiburghouse: So I came up with a program and a plan to scan through all the disks and find these things. The colonel who was in charge of the whole base had to approve running this thing. Because we had to take the computer offline to run this scanning program that I had come up with. But they let me run it. We found all the missing records.

Hendrie: Wow. Did anybody ever find the programming bug that was dropping the records?

Freiburghouse: I don't recall whether that was ever found.

Hendrie: Okay.

Freiburghouse: But we did get all the records back.

Hendrie: Okay. Wow.

Freiburghouse: I spent two years on that assignment.

Hendrie: So you went to the training. And then, you went with the machine...

Freiburghouse: Back to North Dakota.

Hendrie: Back to North Dakota. Got it.

Freiburghouse: While I was at North Dakota, I had the night shift at the air base. I went to the University of North Dakota and got a job in the computer center. Because they needed somebody to run the computer there. The IBM technician who was responsible for the computer on the base was also responsible for the computer back in the University. He told the University about me. And so I got hired at the University to run their computer, which was a 1620. It was IBM 1620, which was a character oriented, quote, "scientific computer." It was comparable to the 1401, 1410 series, which was character oriented, but intended for business applications. And the 1620 was more intended for what they called scientific applications, because it had hardware floating point. In any case, that gave me a brand new transistorized computer...

Hendrie: A brand new thing to play with.

Freiburghouse: It was the first time I'd seen a high-level language. It had a FORTRAN [IBM Mathematical Formula Translating System] compiler. And that was, like, wow. Because we had to program the 305 in machine language. Not assembly language, machine language.

Hendrie: Oh my goodness, there wasn't even an assembly?

Freiburghouse: No. I...

Hendrie: So you had coding sheets where you put in one's and zero's?

Freiburghouse: No, no. It was character oriented. So you used the op code as a character. And the addresses were characters...

Hendrie: Got it. Okay.

Freiburghouse: But it was not symbolic. So there wasn't an assembler...

Hendrie: <Inaudible>.

Freiburghouse: I wrote an assembler for it. But the machine was so slow that it just took forever to assemble a program.

Hendrie: Yes, okay.

Freiburghouse: Anyway, FORTRAN and the idea of programming in a high level language just took me. And that was it. That was what I wanted to do.

Hendrie: Forget history, the politics...

Freiburghouse: Yes, because I was just fascinated with computers and the ability to program them in this kind of a language or better. I realized that you could do a lot better than FORTRAN. But that's what really started me down a path, or a career, in programming. In fact, specializing in programming languages and compilers. My next assignment, the Air Force sent me to San Antonio. It was the headquarters of the Air Force Security Service, which was really part of the Air Force that did data processing and more for NSA [National Security Agency]. So at that time, the Air Force was flying missions around the perimeter of the Soviet Union, collecting signals of all kinds. Trying to locate radar bases and other interesting things. We would process that data to try to identify where these sites were

and what they were doing. So there was a lot of computation involved in doing that. That's what we were doing back at the base in San Antonio.

Hendrie: So that wasn't done at NSA headquarters?

Freiburghouse: Well, I don't know what they did. We weren't told what they did.

Hendrie: <Inaudible>.

Freiburghouse: We weren't even told all that we were doing.

Hendrie: Now, do you have to have...

Freiburghouse: Yes. I had a top secret cryptographic clearance in order to work there. Although, frankly, I didn't really know a lot about what I was doing. But it was a secure facility. We had a huge map of the Soviet Union. And we had flights. All the planes that were flying in the Soviet Union were tracked on our boardroom board. So it was a pretty interesting place to work. I did a lot of programming there. We used FORTRAN. And we had a 1401 and a 1410, which weren't really powerful enough to do what we needed to do. We had limited access to a 7040, which was a 36-bit IBM scientific machine. We ran programs there. But the versions of FORTRAN on these machines were different. So the program written to run on the 1400 series wouldn't compile and run on the 7040 series. I actually took the IBM FORTRAN compiler that IBM had produced for the 1401 and modified it. So it would support the same language, the same source language that was used on the 7040. And I did that on my own, sort of...

Hendrie: Oh, you said, "This is something that would be very handy to have."

Freiburghouse: This would help us get our job done. So I was, again, a hero for that. It was a chance to actually do a compiler. Now, I started with somebody else's compiler to do it. But I had to dig into it, see how it worked and make significant changes to it to get it to do what we needed.

Hendrie: Yes, there were no courses in compilers or anything like that...

Freiburghouse: No, no, no...

Hendrie: No books or anything to read.

Freiburghouse: No, no, no. Then, these compilers were written for a really small computer. We had eight kilobytes or 16 kilobytes of memory on this thing. So the whole job was to figure out how to get this done in a small amount of space. Anyway, I did that. I got to go to the Pentagon to run programs in the big computer that was housed in the basement of the Pentagon. After we were able to run them on the

equipment that we had access to in San Antonio. I got to fly on the general's airplane to the Pentagon to do this work.

Hendrie: Okay. <Inaudible>.

Freiburghouse: I had three [years]. I think it paid me $125 a month. Plus, free food.

Hendrie: So what did they have at the Pentagon that was...

Freiburghouse: It was a 7090 or a 7094. It was the biggest and best that you could get your hands on. I think my superiors ran the same program at Fort Meade that NSA had. But I didn't go there. In any case, at that point, my tour was up.

Hendrie: Was this a three-year...

Freiburghouse: It was a four-year tour. The people that I was actually working for in San Antonio were civilians. They were GS employees of the government. They were scientists and math people. They wanted me to join them, and offered me a GS position that was a really high position. If I look back on it, I probably should've taken it. I would've retired a long time ago. But I didn't. I was eager to go out and take over the world. I left. I went back to the University of North Dakota, because they offered me a job. I still didn't have a bachelor's degree.

Hendrie: You just had one year...

Freiburghouse: Well, I'd taken some courses in San Antonio, but I still had less than two years of college. I wanted to finish. Even though I didn't need to get a computer science degree, I pretty much already had done what I would've done with a computer science degree. But I wanted to have a degree in something. I wanted to finish my history political science degree. Anyway, I went back to the University of North Dakota. I was a freshman. I remember I took college Algebra and some other courses. I ran the computer center. I taught a course in computer programming. I mean, it was just way too much. It was just burning the candle at all three ends. I mean, just impossible. I was being recruited at that point by Honeywell, located back east in Newton, Massachusetts.

Hendrie: Now, they located you?

Freiburghouse: They were trying to sell a computer to the University.

Hendrie: Oh, okay.

Freiburghouse: Well, I don't know if I was being recruited, as I was, kind of, using them to get somewhere. But I had an interview, I think, initially in Minneapolis and then, later in Boston. I had another interview at Cornell University. I had job offers, I think, from Cornell and from Honeywell. But I decided Boston was a lot cooler place to live than Ithaca, New York or wherever Cornell's located.

Hendrie: <Inaudible>.

Freiburghouse: So I moved to Boston to work for Honeywell. My job was to write a FORTRAN compiler for the Honeywell 200, 2000 series computers. Which were very similar to the IBM 1401, 1410 series.

Hendrie: Yes, they were selling them as knockoffs for the 1401.

Freiburghouse: Yes, they weren't completely compatible. But they were very, very similar. Since I'd done a compiler or reworked a compiler for the 1401, it was a piece of cake to do it for the Honeywell equipment. I did that. I was part of the team, a small team, I think, of about six or eight people.

Hendrie: Who did you work for?

Freiburghouse: Oh, Marty Greenfield.

Hendrie: Was Dick Clippinger there?

Freiburghouse: He was there. But I didn't work for him. I met him. Anyway, we finished that project, and I left to go to work for General Electric at MIT [Massachusetts Institute of Technology]. Another one of my coworkers had left and gone over there and told me about what an exciting place it was. He had graduated from MIT. So I went over there. It was the Multics project, which was a successor to the CTSS [Compatible Time-Sharing System] . But it had a much broader vision. It really was Cloud Computing way ahead of its time. The whole notion was that this was a computer utility that, as a user, you simply paid for whatever you used, in terms of storage or computing time or connect time. It was a little more than the initial concept of a timeshare system. But it was a glorified timeshare system.

Hendrie: Now, at what stage were they in the development of that when you went over there...

Freiburghouse: I went over there because they had gotten into serious trouble with the lack of a PL/I [Programming Language One] compiler. The plan for this system was to write it in PL/I.

Hendrie: Write the whole system?

Freiburghouse: The whole system, everything was to be written in PL1. They initially had given a contract to develop a PL1/I compiler to, I believe, Computer Sciences Corporation. Computer Sciences

had failed to produce a compiler. Meanwhile, Bell Labs had produced an interim compiler, called EPL, Early PL/I. Which was a subset dialect of PL/I, but it worked. But it produced horribly inefficient code. So it wasn't suitable to build a production environment. You could build the initial prototype software with it. But the quality of the code was poor and so inefficient that you couldn't really build a final system.

Hendrie: Yes, you couldn't build an operating system, for instance?

Freiburghouse: Well, they did. But it could be a dog, and it was.

Hendrie: Yes, it was a real dog.

Freiburghouse: Then, there were problems with EPL. So, initially, I came in...

Hendrie: <Inaudible>. Were they there or was it written at Bell Labs?

Freiburghouse: It was written at Bell Labs. It was by Doug McIlroy and Bob McClure. But they didn't really want to pursue this. This wasn't their thing. They did it because they needed it. The partners in this project at that time were Bell Labs, General Electric and MIT. The funding came from ARPA [Advanced Research Projects Agency]. So General Electric said, "We'll take over responsibility. We'll build a PL/I compiler." I was hired to be part of that project. I became the team leader of that project. I wasn't initially the team leader, but I became one. We designed and built a full PL/I compiler that produced optimized object code.

Hendrie: How long did that take?

Freiburghouse: Not very long. But I don't remember, precisely. There is a paper that I wrote about that, which is available online. I have a copy of it. I gave this paper at the 1969 Fall Joint Computer Conference in Las Vegas. It's the first time that I gave a paper. It turned out there were, like, 3000 people in the audience. It's where they have the prize fights and all that kind of stuff.

Hendrie: Oh my goodness.

Freiburghouse: TV lights in my face. There was this kid up there on the stage.

Hendrie: A sort of scary sight, yes.

Freiburghouse: It was. But I pulled it off.

Hendrie: So you're still in your 20's probably, aren't you?

Freiburghouse: Yes in '69. I was born in '41, so yes.

Hendrie: Okay.

Freiburghouse: We did a second version of the compiler. Just because second versions are always better than first versions. It was a full implementation of the PL/I programming language. The only other full implementation I believe that has ever been done, outside of IBM. I left— well, General Electric, then, was bought by Honeywell, at some point. So I ended up working, once again, for Honeywell. And I left Honeywell to start my own company, Translation Systems. Because I got the notion that you could make money selling compilers. I had enough experience building compilers that I figured out how to do it in a way that you could separate the target machine from the source language. So you could build a compiler and three-quarters or so or more of the compiler was machine independent. Only the last portion, the code generator knew what kind of target computer it was compiling for.

Hendrie: Now, had you done that with PL/I?

Freiburghouse: I hadn't done it at any of the previous businesses.

Hendrie: Yes, you hadn't organized, but...

Freiburghouse: No, no. But if I'm going to go into business, this is the way you want to do it. Because this way, you don't have to build a complete compiler for every machine and every language.

Hendrie: Do all the work.

Freiburghouse: ...do the whole thing over and over again. Also, I had to do it in a much smaller environment. I targeted this for minicomputers, which, by today's standards, are tiny. Even then, they were smaller than the big computers that we were using in the Multics project. I developed an intermediate language. I developed a lot of things that enabled me to build these compilers in a very, very efficient manner, and a way to make them very easy to maintain. So when I went to companies and gave the sales pitch about why you should buy a compiler from me, I was pretty persuasive. I was able to sell or license these compilers to almost all of the major minicomputer companies.

Hendrie: Well, tell me a little bit more about how this business got started. Didn't you do the first one for Data General?

Freiburghouse: Yes. The first minicomputer company that took my sales pitch was Data General. They wanted a version of PL/I for the Eclipse line of minicomputers. They were a hard company to do business with, because they didn't want to give me the rights to my work—they wanted all the license rights. So we had a long negotiation.

Hendrie: Oh really? So they wanted to own the whole...

Freiburghouse: Well, I mean, I was sort of pioneering something here. In that, normally, up until that point, a contractor would agree to build some software for a client. And the client would own the software. The notion that you could some how license this and still retain rights to it was something new. So it was difficult concept for everybody to finally grasp.

Hendrie: Now, who was the person at Data General that you worked for?

Freiburghouse: I don't remember who it was. It wasn't Bill Foster, because he came in the picture later. But we managed to do the deal, get the lawyers to do this thing. Such that I walked away with the rights to my work and Data General had the sole rights to the version that ran on their equipment. That was the model that I used in all my subsequent dealings. So if I could do another one for another minicomputer company, they owned the version that ran on their equipment. But I still retained the rights to do it on other people's equipment. And because of the way I had designed this, the amount of work that I had to do for each new computer was fairly limited. So it was a good piece of business.

Hendrie: Do you remember how much Data General paid you for that first...

Freiburghouse: No, I don't. I know I got a computer as part of the deal.

Hendrie: Well, that was very important, wasn't it?

Freiburghouse: Yes, it was.

Hendrie: To your business file.

Freiburghouse: And those computers weren't cheap.

Hendrie: Yes, exactly.

Freiburghouse: So I got a computer. And I got, I don't know how many hundreds of thousands, or tens of thousands. It wasn't that much—I got a lot more in subsequent deals. I think the typical deal was, like, a quarter of a million dollars in cash, plus a computer, at least on loan for some period of time.

Hendrie: Yes, you didn't need a computer to do the work. But you certainly need one to test the results.

Freiburghouse: Yes. So I had an office in Cambridge, Massachusetts, with two or three minicomputers in it at any point in time. I would be working on, maybe, two different projects at the same time. I was doing all this work myself.

Hendrie: You didn't have any employees at this point?

Freiburghouse: No. I had a woman that I hired, briefly, to do some documentation. I had some contract workers, Bob Frankston, who was later partnered with Dan Bricklin. He was a contract worker. I had another one named David Levin…

Hendrie: Okay.

Freiburghouse: David built a front end for FORTRAN that worked with the same code generator used for PL/I. And I did a subset version of PL/I and a full version of PL/I.

Hendrie: So you could buy...

Freiburghouse: There was a lot of work.

Hendrie: Yes, a lot. So who was your second sale?

Freiburghouse: I don't remember the order. Prime Computer was a good contract. I think it may have been the second one. I had another contract with Digital. But Digital did their own code generator. They licensed all the stuff in the front, did their own code generator. Control Data Corporation did the same thing. They licensed the front and did the back.

Hendrie: <Inaudible>.

Freiburghouse: It was for their big computer. I don't remember...

Hendrie: Oh, their big computers?

Freiburghouse: Yes. Raytheon had a minicomputer, which I think never went to market. They did the same thing. They licensed the front and did their own back. Well, Machine Bull in France licensed everything. They did their own back.

Hendrie: So all of the optimization and all of that was still in your...

Freiburghouse: In the front.

Hendrie: In the second half of it, once you got it into an intermediate—is that correct? Or how did you structure your part?

Freiburghouse: The initial stage was lexical analysis. That was, basically, a finite state machine, table driven. The next stage was parsing, which was also done the same way. It was a top-down or cursive descent parser, but it was driven by a finite state machine. It would process all the declarations. And then, what I called semantic processing. Where you go over the intermediate language and pick up the connection between a symbol and its declaration and plug in whatever kinds of data conversions are required because you're adding a floating point to an integer. But that's all in the intermediate language. Then, you're still looking at intermediate language, which is independent of the target computer. Optimization phase went over that and found common sub expressions and eliminated them. I have a paper, which I published, which talks about how to do that. It also talks about how to allocate registers in the code generation stage, in a very efficient manner. So that paper shows what the intermediate language looks like. But it's really a paper that's talking about allocating scarce resources, namely machine registers. Based on the usage information that you could collect from the optimization stage. Anyway, that paper has been referenced a gazillion times since I wrote it. I wrote it in 1974. It's still being referenced today. I have a copy of that. Last I knew, it was required reading or recommended reading by the computer science students at MIT.

Hendrie: That's good.

Freiburghouse: So I'm proud of that.

Hendrie: Yes, cool.

Freiburghouse: Anyway, I was very active in the standardization of PL/I. In particular, the standardization of a subset of the language, called G, for General purpose. Because PL/I was a huge language with all sorts of bells and whistles and things in it that really weren't necessary which made it much more difficult to implement and use on smaller computers, like minicomputers. But made it more difficult to use for everybody.

Hendrie: Yes, it wasn't just more difficult to implement, but it was hard to use.

Freiburghouse: Yes.

Hendrie: Because it was just too complicated.

Freiburghouse: Well, and it also was very permissive. I mean, you could write anything and it would by default, interpret things, and by default, make declarations of things you had forgotten to declare. A lot of things that I didn't think were safe programming practices.

Hendrie: You could be very sloppy?

Freiburghouse: Yes. In any case, I led the effort to get it standardized, both in the United States and internationally. It was both an ISO [International Organization for Standardization] standard and an ANSI [American National Standards Institute] standard. Through those organizations, I met a lot of interesting people from other companies. That's how I met Gary Kildall from Digital Research.

Hendrie: So I don't think I quite got the date, if you remember when you started Translation Systems? Did you tell us that?

Freiburghouse: I think it was '74, 1974.

Hendrie: Okay.

Freiburghouse: I had finally finished my work at—I graduated from Northeastern with a degree in history and political science.

Hendrie: Oh, so you kept moving your credits to where you were?

Freiburghouse: I did all this. I went to school at night and finished my degree in history and political science, which I never used for anything. But it was an accomplishment, and I checked that off...

Hendrie: You had a college degree now. Yes, exactly.

Freiburghouse: I actually gave lectures at MIT and Brown and Bell Labs and GE Labs [General Electric] before I had a degree. But I got the degree. I'm not sure I gave any lectures after the degree.

Hendrie: Well, that's pretty good.

Freiburghouse: In any case, the compiler work with Translations Systems was technically fun. I made money at it. I don't think I was a very good businessman. Obviously, Bill Gates did a lot better than I did, even though I had a lot better technology than he had. So, obviously, I wasn't as good at parlaying this into a real pile of money as I could've been, or probably should've been. I was just more focused on the technology and being able to sell this thing all over the place, like an evangelist.

Hendrie: You enjoyed doing it...

Freiburghouse: Yes. I enjoyed doing the work. I still enjoy doing that work.

Hendrie: Okay.

Freiburghouse: Two weeks ago, I taught myself C# [C Sharp], and I wrote a program that plays tic tac toe. You can't beat it. In any case, my early customer, Data General, contacted me. Bill Foster was now the VP of Software there and he was interested in starting a computer company, because he had known the people at Tandem. He had worked at Hewlett-Packard. The guys who started Tandem had worked there. He knew them. He saw their success with Tandem Computer. He was motivated to do something like that. So he came to see me about licensing compiler technology for his computer company, as yet to be. I said, "Well, show me the business plan." Because I was interested in doing something more than what I was doing. So that's what started the whole ball rolling for Stratus Computer. Because I looked at the plan, and I saw, well, I didn't know much at all about Tandem. But the idea that you could put together some computers and make the whole thing fault tolerant was exciting. What Bill had initially put down on paper wasn't very different than what Tandem already had. So I thought, well, that's not going to cut it. But there are other ways to do this that would be significantly different. And I thought if we had hardware that didn't fail; the whole programming problem would be greatly simplified.

Hendrie: Yes, greatly simplified.

Freiburghouse: That would be a tremendous selling point and a product differentiator. I convinced Bill that that's what we should do. But I didn't know how to design a piece of hardware that would accomplish this. I knew it could be done, but I wasn't a circuit designer. I knew I couldn't do that part of it. And what would make my job easy, which was to do the software.

Hendrie: And not have to have—

Freiburghouse: Well, yeah, the whole problem with the Tandem approach was it made the programming extremely complicated. If it wasn't done correctly, you didn't have a fail-safe operation. You might have thought you had one, but you didn't. So that's when I started talking to Gardener Hendrie and other people. The whole ball started rolling to get Stratus Computer off the ground.

Hendrie: All right. That's very interesting. I hadn't heard the part that you were the person that suggested making the hardware fault tolerant.

Freiburghouse: Yes.

Hendrie: Well, that's cool.

Freiburghouse: I mean, I thought three-way voting would be the obvious way to go. But I wasn't a circuit designer...

Hendrie: That was my initial take, too. It was very obvious to me you could do that. So it was doable. But then, when we got into it, we came up with the conclusion there would be a better way.

Freiburghouse: Well, and the better way was a lot better, because you could pull the failed component out.

Hendrie: Exactly.

Freiburghouse: The whole thing was hot pluggable. I'm not sure, but I believe that the Stratus machine was the first or one of the very first machines where everything was hot pluggable. Now, you buy a server from Dell or anybody else, and all the stuff is hot pluggable.

Hendrie: Yes, that's just how it's made.

Freiburghouse: Yes.

Hendrie: Yes, it's just ordinary technology. Well, I remember when Bill talked to me about the possibility of joining. He said, "Now, I've got this guy, Bob Freiburghouse," —and I did not know who you were— "you ought to meet him." And I remember meeting you. Then I don't think I told Bill, but I remember saying to myself, "Well, it was obvious in about 30 nanoseconds that the software was going to work on this computer." I had worked on too many machines where the hardware was up and running and the software didn't work. It didn't work. [It was] just agony to get the software to work. And oh my goodness. Well, that's one of the risks of the startups that isn't going to be in this one. So that probably was a big influence. Anyway. This is your oral history. So we're mixing up stories here.

Freiburghouse: The big challenge, really, with Stratus Computer was putting the operating system design together and the team and getting it done.

Hendrie: Yes.

Freiburghouse: I had worked with a whole bunch of mini computers in the Translation Systems business and looked at all their operating systems, which were very similar. But at that time, of course, everybody was proprietary. So everything was a little bit different. What I pulled together for those application program interface spec level of the VOS [Virtual Operating System] for Stratus was an amalgamation of what I had seen with these others, most heavily influenced by VMS [Virtual Memory System] from Digital since I thought it was probably the best of the breed at that time. So that's where the application program interface concept came from. I wrote out all the specs of what this operating system will do from an application perspective, and then tried to dig down in terms of the design of some pieces of it, but not all of it by any means. What I wanted to do was to control that interface, and not let the developers just keep on embellishing it, because that's how projects never finish, because they don't build to the spec. They keep adding on more and more stuff to the spec. So as I put the team together I was really hard on the notion that this is what we're going to do, this is the time that we have to do it in. We're not going to change it. We're not going to embellish it. We've got to do it, and I rode herd over it.

Hendrie: This is an implementation not an invention project.

Freiburghouse: Well, that's right, and you could be creative in how you built it, and our objective was to build it rock solid because it wasn't going to do any good to have full tolerant hardware if we've got flaky software. The vulnerability of this whole concept really was running one piece of software on there, and if you got errors in the software it'll crash like any other computer would crash. So the idea is to get this core kernel operating system rock solid, and do that by keeping it as simple as possible, but still meeting all of the necessary objectives from what has to support transaction processing and timeshare like usage. Only the part that I didn't anticipate or understand very well was the need to have networking, and we had to invent all our own networking. There weren't any standards, and today there are all sorts of standards. We had to do everything. We had to do our own word processor. We had to do our own communications software. We had to do all of it, and we had to do it all by delivery date, otherwise the customer didn't have enough software to do their job. So we had to deliver compilers. We had to deliver application software, and word processing, and editors, and all the communications software. That was a huge, huge undertaking.

Hendrie: So how did you pick the people that it turned out they could do this?

Freiburghouse: They could do it, and fortunately the Boston area had a lot of talent, and I was pretty well plugged into that talent because I had worked with some of the best people in the various minicomputer companies, as well as the team back at MIT that was working on the Multics project all those years. That project was going nowhere, and the people were ready to move on to do something else, and there were a lot of very good people. They believed in me, so they would believe my pitch that this was a big opportunity.

Hendrie: And this is going to work.

Freiburghouse: This will work and, "Hey, what have you got to lose? You're young. You're employable."

Hendrie: Yes, "You're really good at what you do."

Freiburghouse: You don't have to move.

Hendrie: "You can get a job at this..."

Freiburghouse: "You don't have to move to California. You don't have to go anywhere."

Hendrie: Yes "Come and try this."

Freiburghouse: Yes.

Hendrie: Do you remember when you were trying to figure out who was going to do the operating system? Do you remember approaching Dave Cutler?

Freiburghouse: I worked very hard to get Dave Cutler to come to do the operating system because I figured he was the best that I had met. He had done the best operating system that I knew back at Digital. He was a designer of VMS, and he was the lead person on the PL/I project at Digital.

Hendrie: Oh, I didn't know that.

Freiburghouse: That's how I knew him. He did that because he was sick of doing operating systems. So when I worked on him to get him to come to Stratus he just didn't want to do another operating system. So I eventually had to give up on him.

Hendrie: You had to give up on him, and eventually he did do another operating system.

Freiburghouse: Yes. He went to Microsoft and did NT.

Hendrie: Yes, exactly, all right. Well, that's interesting. Talk to me about your darkest moments during the development of the software at Stratus. Were there any times that things didn't go well for a little bit, or there was trouble that you didn't anticipate?

Freiburghouse: Yes, I don't remember precisely, but there was at some point when we discovered that the damn thing wouldn't take a page fault properly. Whatever we thought we were going to do to handle page faults wasn't going to work, and we had to come up with another way to do it. I believe that was a second processor that we had to put in the system in order to be able to handle the page fault.

Hendrie: I think it was too.

Freiburghouse: Yes, that was kind of a big deal.

Hendrie: Yes, that's true.

Freiburghouse: It happened a lot later in the game than it ever should have. Actually, that kind of problem was persistent all through the history of Stratus where we'd find out there was something about the chip. The biggest problem was indeterminacy in the chip. You've got two chips supposed to be running exactly the same thing and having exactly the same internal state, blah, blah, blah, but they didn't always have that property. Then you have to go back to the chip manufacturer and say, "Hey, look, we need this changed." Well, Motorola, or Intel, or whoever is not too keen on changing the chip design for a low volume user like Stratus Computer, so that was a big problem.

Hendrie: So how long did it take, and how many people? Do you remember how many people and long it took to shipment of that system?

Freiburghouse: God. I'm guessing now. I think we had about 15 people in software engineering for the whole team. That included the operating system, and compilers, and communication software, and probably at least one, maybe two people doing editor. So, yes, I think about 15 people with a year and a half or so of development time. I remember this guy from Honeywell came over. I think he had it in his head that we somehow had stolen something from them. He wouldn't come out and say it, but I knew that that's what he was thinking because how in the world could these people build something so complete so quickly if they didn't steal it. He came over and we gave him a presentation on the design and what we were doing and he left realizing that we weren't building a copy of Multics. They thought that's what we were doing because we had a lot of Multics engineers, but our system didn't look much at all like Multics. Even at the core fundamentals it wasn't a segmented system. It had paged memory, but so did a lot of other people. The API [Application Programming Interface] didn't look anything like Multics.

Hendrie: Just the basic commands that _____, all right. So after that system got shipped what did you do next at Stratus? You stayed at Stratus for a while.

Freiburghouse: Yes. Well, everything changed. Once the company became successful it wasn't a matter of managing a software development project and supervising implementation. It was a matter of trying to manage the growth, hiring people at the rate you needed to hire people, trying to keep the sales force from selling stuff we didn't have yet. It was a whole different set of problems.

Hendrie: Yes, they were not technical problems anymore.

Freiburghouse: No. No. They were managerial problems.

Hendrie: Yes, okay.

Freiburghouse: I think the problem that I've always had is interfacing with sales-force and understanding sales -force motivation, and their lack of understanding of us, us being the engineering community. I think that's particularly true in a situation where it's almost a semi-custom type product, or at least the sales people treat it as if it is. So they can go out there and sell what the customer wants rather than what's in the brochure. Now I think that era has passed, at least for software and hardware companies. Now it comes in a shrink wrapped container, or it's downloaded and that's it.

Hendrie: Yes, you don't have any options. I should turn this off. So did you continue to manage the software at Stratus? At some point didn't you take over the hardware too?

Freiburghouse: Yes. I ran both hardware and software for some period of time. I don't remember how many years, and then the company was reorganized into sales divisions. Initially it was split so there was

a division of people that sold a System 88, or supported System 88 sales, I guess is the best way of saying it.

Hendrie: You need to explain that to our viewers, please. You may need to back up to explain that, too.

Freiburghouse: IBM approached Stratus and wanted to sell the Stratus computer under an IBM label, and it was pretty much the same team of people that had approached Mr. Gates in that they had come out of the PC [Personal Computer] portion of IBM. So they were kind of the mavericks in IBM that had done the PC business and now they were going to do another end run around the establishment with a System 88. Being a very small company struggling as we were we thought, "Well, having IBM sell our product is going to be a good thing." It turned out, I think, not to be a good thing at all because IBM simply sold it in the same places that we would have sold it, and we could probably sell it better than they could. We were certainly motivated to sell it more than they were motivated to sell it. So there was a conflict from day one in accounts because we'd go into an account and IBM would come right into that same account and try to sell the same thing. Bill Foster decided the solution to this was to separate a group of people within Stratus to support System 88 sales, and try to somehow separate the market place so that this conflict wouldn't take place. One of the markets that was given to IBM was the telecommunications market, namely the big regional telephone companies because AT&T had been split up so we had regional telephone companies that used to be all part of AT&T. Now they were independent companies, but they still ran the landline phone businesses in the regions of the United States. So that was supposed to IBM's turf, plus some other accounts. Well, even that didn't work very well. IBM was not pushing the product the way they needed to push it, and it dawned on us that telecommunications was really the place where Stratus computers should be playing, that that was an ideal kind of application, an ideal environment for our equipment, and IBM wasn't doing enough with it. We began to have success in other telecommunication companies around the world, and within the United States. At that point Bill Foster decided to reorganize the company further, created a telecommunications division, a North American sales division and an international sales division, and I was appointed the head of the telecommunications division. It was a wonderful job in the sense that it wasn't just sales. We had engineers that could build specialized products for that industry, build software. Not applications, but like the communications software and other specialized middleware that was appropriate to that industry. We could sit down with the technical people at NTT [Nippon Telegraph and Telephone], or any of these big companies and workup a joint project, a significant project. That was completely different than a field office driven sales activity. That had to be a headquarters driven, headquarters invested activity, and I understood that, and I organized the division that way. I had a team of really competent technical people that understood the telecommunications business, engineers, and those resources would then work with the sales team to secure a big, big contract. We were really successful. I mean, we also got as part of that division the customer service group, and so we supported the installed base of machines everywhere, including all the ones that IBM had sold. That included ones in the telecommunications companies.

Hendrie: That IBM had sold.

Freiburghouse: Yes. It's now Verizon, but if a Verizon account was having a problem it was our phone that rang. I reorganized the way that service was monitored. When I took over service was monitored by how short the call queue was. So their motivation was to get the call out of the queue. I said, "Wait a

minute. That's not the motivation. The motivation is to make sure the problem's been solved. There's a difference."

Hendrie: Yes. And the customer is happy.

Freiburghouse: Yes. The only way you know the problem's been solved if the customer tells you it's been solved, not that you took it out of the queue. I made that change, and I had a really good person running that group. I think we delivered excellent customer service. They were happy, and they were really pleased, and it was a big piece of our revenue, so it was important to do properly. In any case, the telecommunication division was the growth division of Stratus at this point, because in the traditional markets we were getting our lunch eaten by servers that were networked. You could get a couple of SUN servers, or a couple of whatever servers, or a network of servers, and they could provide very high level of availability at very low cost, and it was very clear that that was the future. I mean, our computers were much more expensive, took much longer to develop a new hardware product because it was much more complex. Not only did it have a lot more circuitry in it, it had to behave in a really precise way. It was hard to do. In any case, the other sales divisions were not at all happy with the success of the telecommunications division because they saw that their business wasn't doing all that well, and they kept seeing telecommunications companies in their quote "territory" were doing business with my division and not with them, and they were losing revenue from that. But that was the mission. That's what we were supposed to do. But that created a problem at the top of the company because they would come and complain that they should have had this account, or they should have been doing that.

Hendrie: Yes, that business ought to be theirs, and it never got resolved effectively.

Freiburghouse: It didn't get resolved and the solution that was imposed was across the board cuts, financial cuts, budget cuts, across all divisions without regard to what divisions were actually producing the growth and the revenue. Then the management of the company was changed and I was out the door, so that was the end of that story. Kind of a bitter end from my perspective, but it was history. I think the company could have done a lot more had it really focused on a core strategy and stuck with it, and telecommunications was obviously a thing that could have been taken a long way had it been grasped earlier and more completely.

Hendrie: In terms of growth.

Freiburghouse: Yes, and long term success. Whether there'd still be duplexed hardware being successfully sold I'm not sure. Probably not because you can network these things so effectively now, and they're cheap to produce.

Hendrie: Cheap to produce and use N+1 strategies for fault tolerance, and achieve it by just switching processes from a failed machine to another one. It can be done without fault tolerant hardware now and without checkpoint.

Freiburghouse: Yes, right. I mean, we saw this. We saw this coming. We saw this really early on, but it's very hard for a company once they're embarked on a particular path using a particular technology to change directions, even when they know better because so many people in the company are invested in what they already know, and already do, and are already making money at. So it's very hard to steer the boat in any other direction. You'll see that a lot with companies.

Hendrie: There are those who say that Microsoft is in the same boat today.

Freiburghouse: On the CNBC [Consumer News and Business Channel] stock channel that I watch all day long everyday they talk about this dead money. That's where money goes to die is companies like Microsoft, Intel and Cisco. They're like the dead technology companies where money goes to die.

Hendrie: So you left Stratus. What did you do next?

Freiburghouse: Well, I worked on a startup that was trying to do healthcare software that would enable chronically ill patients to communicate with their healthcare provider using a customized interface. So that let's say a doctor could set up a set of parameters for me as a heart patient saying, "All right, here's what I [want] you to do at home-- these are meds I want you to take. Here's how often I want you to take your blood pressure. Here are the things I want you to look for," and if I kept doing those things, and I could communicate with him through a personal computer, he wouldn't have to see me. But if one of my parameters was out of range he could then say, "Bob, you've got to come into the office. I need to check you out." Diabetics and chronic heart conditions account for the vast majority of healthcare expenditures, and yet doctors really do not have much choice when scheduling those patients. You come in and he'll say, "Well, I'll see you in three months. I'll see you in six months." He doesn't need to see you in three months. He needs to see you when something goes wrong or is out of range.

Hendrie: And something happens, yes.

Freiburghouse: That's when he needs to see you. It might be tomorrow. It might be six months from now, and so this whole system was to set up to do that kind of thing. So I was really excited about that. I said, "This is great."

Hendrie: It makes a lot of sense, right.

Freiburghouse: This makes sense. This is the way technology needs to be used, and the thing can be made simple enough that anybody could do it at home. If you can't do it yourself, your wife or some caregiver at home could do it. You don't need a nurse to do these things. But we never got the funding for that. At the time when we talked to venture capitalists they were in a mode where they wanted a short payback. This was the beginning of the dot com era where you're going to make a gazillion bucks in six months or a year, whereas what we were working on was going to require a long proof of concept, and it might require a long period for people to sort of change the mode of healthcare delivery. So it was not something that was going to blossom overnight. It just is something that had enormous potential long term. So we didn't get the money that was needed, and so I left because my deal with them was, "We'll

work on this for a certain period of time. If we don't get the money then I'm going to go." Subsequently, I went to New Jersey. There was a spinout of Lucent Technologies called CommVault Systems. There was a group of engineers at AT&T, subsequently Lucent, that had developed some data management software, backup and recovery software that had really been designed for their own labs for back up UNIX systems, or networks of UNIX systems, and backed it up onto optical storage devices. Lucent wanted to spin this out because it didn't fit with the rest of their business. So the plan was to take it private, and take it commercial. I went down there to New Jersey to run engineering and customer service, which was also manufacturing in the sense that they had it because they sold the hardware and the software together to do this. The man who ran it was a long time AT&T hardware engineer. He'd never worked out in business. He'd never run a company, and he had a group of engineers that had done this because they were doing it for their own labs. Well, the first business plan that was put together was not very solid, and of course we didn't make it. We didn't meet the plan. Sales fell short so I couldn't hire the engineers that I had budgeted to hire, and so I couldn't deliver the products that I had proposed to deliver, and everybody was pointing fingers at everybody else. The board replaced the CEO, who in turn replaced me and the head of marketing and sales. He replaced me with his brother-in-law, so I didn't even have an opportunity to defend my position. So that ended that. I was pretty discouraged by that whole process.

Hendrie: What ended up happening to the company?

Freiburghouse: Well, they did well at the end of the day.

Hendrie: In the long run.

Freiburghouse: In the long run they did well. They couldn't go public because the dot com bubble had burst, so the stock that I had was not worth anything, but I kept it. Ten years later they did go public. They built successful product eventually. They bought other products. As far as I know they're doing quite well, and the man who took over is still running it. He's got somebody else running hardware engineering.

Hendrie: Not his brother-in-law anymore.

Freiburghouse: Well, actually a guy who was one of the team leaders that was there when I was there is running it.

Hendrie: Oh, he's running that.

Freiburghouse: Yes. Yes, so they've done well. It was a long road for them, but they stuck to it.
Hendrie: Well that's good.

Freiburghouse: Yes, yes, very good. They canned the original business a long time ago. I mean, they don't sell hardware and they don't sell any of those old products. That was our plan as well. It's just we couldn't execute that plan.

Hendrie: Yes, you couldn't get from here to there.

Freiburghouse: We couldn't get from here to there. We were buried in problems with the original product and not meeting the plan, so.

Hendrie: Yes, so is...

Freiburghouse: Oh, so then I was really discouraged having been basically out of two companies in a period of three years, and I knew that we were going to have to move from New Jersey to go anywhere that I wanted to go, to I wanted to work. I had a job offer from a company in New Jersey, but it was a hardware management job and I wasn't really suited for that. I know that they wanted the VP of hardware to be hands on involved in hardware design, which I knew I couldn't do. So I said, "No. I'm not going to take this." They offered me the job, which was stupid of them, but I said no. There were other jobs that I really thought I was very well qualified for, and really wanted to do, but I didn't get. So it was discouraging. I didn't stay with that long enough. I'm too impatient when it comes to that. If you're going for a general manager or a senior VP job it takes a long time to land it, and I just didn't have the patience for it. There was one with Cognos in Canada that I thought, "Oh man, this is perfect," but I didn't hear from them again, so. There was another one later after we had moved to the Caribbean, there was another one in Boulder Colorado, but that company ended up being bought very shortly afterwards. They'd actually offered me the job, but then I didn't get the written offer. So, "What's going on?" Then weeks later they were bought by somebody else, so I think that was behind it. In any case I said, "All right, enough of that," and I got a job as a technology director for a small school in the Caribbean. I taught AP [Advanced Placement] computer science, and I wired the campus, and I learned how to be a system administrator for NT and Windows Server and all that junk, which I didn't care for at all, but it was a challenge, and I did it, and I did a good job. I kept it all running in worst possible conditions. I mean, this campus is laid out along the ocean with open windows, salt air blowing through the classrooms, blowing through the computers, blowing through the switches, hitting the exposed wiring that's running around outside of the building.

Hendrie: Oh no!

Freiburghouse: Of course you've got a student environment, so you've got hundreds of users on any particular computer, not one person at his desk with his computer. It's a very different kind of environment to support, and you need all sorts of things to lock it down so the kids don't mess it up, and hack it, and do things that are inappropriate for students to do. Anyway, I did that for eight years. I enjoyed the teaching part of it. I don't really enjoy being the problem solver for everybody. That's annoying, but you do what you have to do. But I taught introduction to programming. I used Pascal for that. I still like Pascal as a teaching language. It was designed for that and it's a good language. Then I taught Java because that's required for AP computer science. So I learned Java and I taught Java as part of that. Then more recently I taught myself C Sharp, which is very similar to Java, but a little bit

nicer, and it comes with this great development environment from Microsoft they really did a good job with.

Hendrie: They really did a good job with that one?

Freiburghouse: Yes.

Hendrie: That's good. Let's pause. I want to roll back and maybe you could talk a little bit about those 18 months when the Stratus system—my understanding is that besides managing the 15 programmers that were working at it, you also wrote a Pascal compiler for Stratus at the same time, and you had never written a Pascal compiler. Tell me about that.

Freiburghouse: Well, the compiler technology that we had was the technology that had been licensed from Translation Systems which was my development, my design, and it was initially built so that you could create front-ends or compilers for various programming languages and share a common co-generator, a common backend. So, what Translation Systems had at the time that Stratus started was a PL/I frontend and a FORTRAN front-end. We hired somebody at Stratus to build a COBOL [Common Business-Oriented Language] front-end, and the other language that was...

Hendrie: Otto Newman.

Freiburghouse: We hired Otto Newman, and the other language that was in fashion at the time was Pascal, and we didn't have a front-end for Pascal. So I wrote a front-end for Pascal. Now, Pascal was a beautiful, simple language. It's fairly easy to compile, and I also wanted to challenge myself because I figured I knew how to build the whole front-end as a finite state machine. I had built lexical analyzers as finite state machines, but I said, "I know how to build the whole thing as a finite state machine," and so I did. As a result it was very, very simple in structure, and very easy to debug, and therefore I could do it sort of in my extra time, if there was such a thing, and it worked.

Hendrie: It worked. And it got delivered.

Freiburghouse: Yes. Actually, when I was down in the Caribbean, I did this similar thing, as I had students doing a project where they were building a compiler for Pascal. They wrote it in Java, and I wasn't going to have the students do something that I couldn't do, so I did one along side of them partly so I could show them what they needed to do, and partly just because it was a challenge. I did it the same way that I had done that one many years ago at Stratus. It stretched my old brain to figure out how to do it again, but we did it.

Hendrie: All as one big finite state machine?

Freiburghouse: Well, yes. It was a separate one for the lexical level and another for the next level.

Hendrie: Well, that's really good.

Freiburghouse: Along the way I developed this language called TBL, which if you look in the literature it's referenced. It's a really, really simple language. I have a little paper that I wrote way back when that defines this language, and I used that to build all these compilers at Translation Systems and later on. They were all table driven, and build the tables by writing a little program in TBL, Table Building Language. So it enables you to create a table that will drive a lexical analyzer or drive a parser, but you're not just looking at an array of integers. You're looking at a language that actually reads, so you know what you're looking at, and you understand what it's doing, and you could see errors in it, or you could modify it. And it's pretty simple, but if you choose your names properly and construct this thing it's quite readable. So, that's been referenced quite a bit too because people use it.

Hendrie: So even you used a language of your own invention.

Freiburghouse: Yes.

Hendrie: To write...

Freiburghouse: A compiler for another programming language, yes.

Hendrie: Yes, to write a compiler for another programming language, construct it. Well, that's pretty cool. Well, see we can get into that. You also had some comments while we weren't on camera about later thoughts about your healthcare initiative or that company, and that whole idea of remote monitoring.

Freiburghouse: Yes, I mean, one of the ideas that really excited me my whole career was this notion of being able to do healthcare delivery in a much more efficient way than we do now. Today most of healthcare costs are from chronic diseases like chronic heart conditions and diabetes. Those are the patients that spend the most amount of money in the healthcare system, other than the last two weeks of everybody's life when you go to die, but let's skip those. Because today the delivery is very, very inefficient. You go see a doctor, and he does all his checks, and gives you your prescriptions and he says, "Well, come back and see me in three months, or six months, or next week,". He doesn't really know when he needs to see you. He needs to see you when something isn't right. When one of your healthcare parameters is out of whack that's when he needs to see you. He doesn't need to see you in six months. He might need to see you next week, so what we really need is a communications mechanism whereby the patient or the caregiver who lives with a patient can monitor the patient on whatever parameters are appropriate for that patient, and communicate that information to the doctor in essentially real time. We have all the technology to do that now especially with the Internet and with the mobile devices, and servers. You could easily set this stuff up so that a doctor could say, "Okay, Mr. Freiburghouse this is what we need to do. This is what you need to do. Here are the things you should be doing with your diet. Here are the medications you should be taking. I want you to check your blood pressure three times a week. I want you to check your weight every week," and that's my job as a patient to go home and do those things, and put them in the computer, or the handheld. If I'm too feeble to do it somebody who lives with me can do it. Then the doctor can come into his office in the morning and he can see, "I've got 500 patients, but this one, and this one, and this one need to come in a see me now,"

or "This one needs to go to the emergency room now." That technology is quite possible. It's quite feasible. There are no breakthroughs required to do that. What is required is a change in the way healthcare delivery works, the expectations of the doctors, or the expectations of the medical facility. The benefits could be enormous. Not only can you do a better job in terms of providing better medical outcomes, you can save a lot of money. So you have a way of saving money and improving outcomes at the same time. The potential's huge. The problem, of course, is it's a major change in the way people behave and deliver healthcare. HMO [Health Maintenance Organization]s and other organizations like that that benefit financially from improved delivery would be the logical implementers of such a thing. A typical doctor's office doesn't gain one way or the other. A typical insurance company doesn't gain one way or the other. They're not delivering the healthcare. They're simply paying for it, but a HMO is doing both.

Hendrie: You said you had somebody research patents and things for you.

Freiburghouse: Oh well, yes. This thing, this idea kept coming back to me saying, "Why hasn't this happened?" Every time I go to the hospital and deal with this stuff I say, "Okay, there's got to be a better way." There is a better way, and why aren't we doing it? So I actually had a patent search done to see if anybody had patented the basic concepts behind this, and there are a whole bunch of patents that have been issued. Unfortunately in all the stuff that I read the key ideas are all wrapped around a whole bunch of other stuff, and as far as I know there are no working application or working products that really do what I described. There are plenty of patents that cover various aspects of it.

Hendrie: So the idea's been thought of, but nobody's gone and...

Freiburghouse: Right, and the people that I worked with way back then, I mean, they obviously had this idea, and I thought, "Well, maybe those guys succeeded eventually in doing something with it," but they didn't either.

Hendrie: So your conclusion is that it isn't a technology problem it's a changing behavior problem which is very hard.

Freiburghouse: Yes, as far as I know nobody's put together a really slick system to do this well, but I don't see any difficulty in accomplishing that. The difficulty is getting some healthcare organization, an HMO, or VA [?], or somebody like that that stands to benefit financially from better outcomes or better delivery to actually implement it. If it were implemented, and then the word spread around everybody else that was in a similar kind of situation where they could benefit financially would adopt it. It would make sense to adopt.

Hendrie: Is there a need for a system like this to be able to do blood work, or to be able to do what is equivalent of what a stethoscope does?

Freiburghouse: No, and there have been people who have gone in that direction, but at least from the work that I did with these other folks that that wasn't necessary. It's just sort of basic things that anybody

can do at home like blood pressure, or temperature, or weight. Are you actually taking your meds? That would be enough to make this very workable. I'm sure there are cases and certain types of diseases where more might be beneficial or even necessary, but even that first step would be huge. Now, diabetics already do their blood sugar stuff all of the time, but they don't send it to anybody. They do it and they look at the number.

Hendrie: Well, that's good to know.

Freiburghouse: The same with home blood pressure. Everybody can do their own blood pressure.

Hendrie: Right, everybody can do home blood pressure.

Freiburghouse: No big deal.

Hendrie: Those instruments are really easy.

Freiburghouse: Now, stethoscope you have to have training to know what you're listening to with it.

Hendrie: Yes. Well, you may be able to record it.

Freiburghouse: Well, that's an idea, yes, actually. You've got to put it in the right place so you're getting the right sound. You can put it over here and not over there.

Hendrie: Well, that's interesting.

Freiburghouse: Yes, it's really exciting.

Hendrie: So that's something that may happen in the future.

Freiburghouse: Well, it better. I mean, healthcare costs are eating our lunch in this country. We've got to do something about bringing it down.

Hendrie: Well, thank you very much for taking the time to do this oral history for the museum.

Freiburghouse: Well, thank you. It's fun reminiscing.

Hendrie: Yes, very good. That's great. Thank you.

END OF INTERVIEW

