Oral History of Thomas A. “Tom” Rudkin

Interviewed by:
David C. Brock

Recorded: March 31, 2015
Mountain View, California

CHM Reference number: X7446.2015

David C. Brock: So Tom, to begin at the beginning, I thought we could talk a little bit about where you were born and when you were born and your family. If you could, tell me a little bit about that.

Thomas A. “Tom” Rudkin: Okay, sure. I was born on March 20, 1951 in Jamaica, Queens, New York. My father had a job that took him there and my mother-- it was my mother and my father and myself at first. I was only about 20 months old when we moved to Cincinnati, Ohio, where I lived for another six years. Started school, learned to play the accordion, played baseball, et cetera, et cetera, became a lifelong Cincinnati Reds fan. My sister was born in Cincinnati. Then we moved to Wichita, Kansas in 1958, where I lived through high school and my parents lived there really to this day, except that we spent eight years in Iowa during that period and went back.

Brock: Where those moves driven by your father’s work?

Rudkin: All the moves were driven by-- my father worked for the same company for his entire career, Equitable Life, and he moved around-- we didn’t move very often. Like I said, he’d been in a couple of places before I was born and he met my mother when he was in Des Moines where my mother was working, because she grew up on a farm in Iowa. My father was a Kansas boy so he had his first job out of college-- he went to University of Kansas, as I did too, after World War II.

And after he finished his bachelor’s in business, he went to Des Moines with Equitable and then Grand Rapids, Michigan and then New York where I was born and then Cincinnati where my sister was born and then back to Kansas, in Wichita where my brother was born. There’s three of us and, like I said, we lived there. After my wife and I were married and we already had our daughter then they left Wichita and went to-- back to Iowa, again for his job, and then when they retired they said, “Let’s go back to Wichita,” so they did. And my mother passed away about four years ago but my father and mother lived in that same house for about 20-some years, not the one I grew up in but a different one that they lived in ever since they retired.

Brock: Did your father have-- what kind of role did he have? It’s a life insurance company. Is that correct?

Rudkin: Yes.

Brock: Or an insurance company in general <inaudible>.

Rudkin: Insurance company, life insurance primarily. Well, so, yeah, he ran an office. Office manager type position. I guess he liked his job. I don’t really know that much about how it worked out but he taught me a lotta things about electricity and plumbing and all those kinds of things you have to do around the house to keep a home going. So I graduated from high school, West High in Wichita, in 1969 and I went to University of Kansas for four years. Wanted to be a chemistry major when I started. I really loved math and science always and I was very good at it. I thought chemistry would be good, then I sort of thought, “Well, maybe physics would be better,” so I started taking a lotta physics classes and taking math all along, of course, but as something I was interested in.

And it wasn’t until I was probably a senior before I said, “You know, I think math is the major.” So I have a B.A. in math. I went abroad my junior year, studied in France for the full nine-month term, in Bordeaux. I had a little French before I-- three years in high school and some courses as a sophomore at K.U. but I was able to learn a lotta French in the first six weeks we were there because this was a group from University of Colorado that had students from Kansas and Nebraska and Colorado and a few other schools. And we had a six week training period, just for this group of 50 students from U.S. first, and that was intensive French language and also learning about the history and the civilization and the culture of France. So after that was over, I felt pretty comfortable speaking in French. I would never call myself fluent but at least I could have any conversation I really needed to have on a broad range of topics. And then I went back, took my senior year at K.U. and graduated in ‘73.

Brock: Could I take you back just a little bit before that? You mentioned that you had always had an interest in math and science. I imagine that was something that you primarily encountered in your school context. Did you have opportunities to explore either of those or technical things outside of school, in high school or before?

Rudkin: I don’t recall doing that. No, I don’t think so.

Brock: Did you have any particular hobbies that seemed associated at all?

Rudkin: No. I mean, I had some hobbies like stamp and coin collecting. I also played golf a little bit. I was never very good. I would bicycle a lot around the neighborhood and then further when I got a little older. But, no, I don’t think I had any science or math hobbies. There really weren’t, that I recall, good out of school opportunities like that. Maybe I just didn’t know about them but I really don’t recall anything like that.

Brock: But music was an interest of yours, is that right?

Rudkin: Well, yes and no. I mean, I had this early training which allowed me to read music. Now, I’m talking about very early, like kindergarten, first, second grade. I sang. I actually sang in a church choir and I sang in the-- there was a school chorus in third grade that they had combined kids from several schools and we would sing in various different places. But, as I got older, I was really not-- well, I guess I was in glee club in eighth grade but that wasn’t really-- that was a class. It really wasn’t much of a public singing type of thing. Maybe a little bit in a school assembly or something. So I wouldn’t really say I was musically inclined, accept that I kind of understood it a little bit.

Brock: When you were in high school in Wichita, were there opportunities for science clubs or anything of that kind that you experienced or did you...

Rudkin: No, I don’t think so. My extracurricular activities-- well, I did debate, in my junior and senior years, where we would go to tournaments around various parts of the eastern half of Kansas. I don’t really recall any, like I say, any outside the classroom kind of math or a science or any really much else. My afternoons after school, other than homework, were playing. Playing with kids in the neighborhood. This was in the early '60s where there were a lot-- I mean, by high school it was late '60s, but all through

the '60s we had lots of other kids in the neighborhood. There were a huge number of kids of course at that time from the baby boom and we would play-- I played baseball for four years in the spring. Bike around. I don’t even remember what else we did. In summer, we had flashlight tag games where a bunch of us, 10, 20, 30 people, whoever, would play flashlight tag in a set of yards that connected without fences in the back. So you could go just about anywhere over several houses worth of land. But nothing really academic outside of school, I would have to say.

Brock: I was just curious about that era when you were in high school, was kind of an interesting, a very full era in terms of American history and changes. I was wondering how you experienced that through the news or in your own life.

Rudkin: Well, it’s hard to remember. Of course, the Vietnam war was going on by the time I was in junior high school. I had a cousin whose husband died in Vietnam and I had some friends who served there later, when I was older, when they were older. Probably the biggest thing was the Cold War and I kind of remember, in 1962 one day, our teacher-- this is the way I remember it. I have no idea if it’s right. Teacher said, “Well, the war will start tomorrow.” What the heck does that mean? I was just scared crazy and I was in second grade or third grade. Fourth grade probably. And that night, when I would hear an airplane I was just-- “What is this?” Well, I’m pretty sure later I put together that this was during the Cuban Missile Crisis and there was concern that a war could break out. But I misheard, misremembered, whatever, what this teacher said, so-- but that was-- I mean, obviously, we were all acutely aware. You had the duck and cover drills or whatever at school. We also had more-- we were more worried about tornadoes, though, than nukes because those were real. Those happened every summer. We may not be near one but they certainly were near enough that you had to know about them. And we had those kinds of drills in school, get to go out in the hallway and get up against the wall and put your head down between your needs and yada-yada.

Brock: As you were approaching the end of high school, obviously interested in math, interested in science, how did you choose the University of Kansas? Did you look at a lot of places or were you focused <inaudible>...

Rudkin: Well, I looked at a few. I applied to Rice and I got accepted with some condition. I had to take the physics-- what do they call them? The College Board. Not the SAT but the subject matter courses that College Board offered. And I scored pretty high in physics even though I’d only just studied the book a little bit because I didn’t have a chance to take any physics. There was a physics class I could’ve taken accept it conflicted with something else. Rice woulda been good but we didn’t really have a whole lotta money and I needed to go someplace which wasn’t quite as expensive, and Kansas was, first of all, a very, very good school. My father had gone there. But, secondly, in-state tuition was very low and so, ultimately, that was why I chose it I think but I don’t regret that for a minute.

Brock: We talked a little bit about the different courses of study that you went through there and ending up with mathematics. Were you able, at the undergraduate level, to form a particular domain of mathematics that you were specializing in or...

Rudkin: No, not at all. My theory at the time, and I still hold this, is that as an undergraduate, at least if you have the means and the brains I suppose, don’t specialize too much as an undergraduate. I took a lot of courses in a lotta different fields. I went to France and took one math class but primarily it was history and language and philosophy. I had taken a lot taken a lot of philosophy even before I went to France and I ended up with-- by the summer after I graduated, I would’ve had enough credits to have a philosophy degree but I used it as a minor instead. I arrived on campus in august of 1969 and I looked at the course catalogue before signing up for my fall classes and I went, “Wow, this is just amazing. There are so many subjects here that I would just really love to learn about,” and so I took 15, 18, even once, 21 credits a semester. Actually, I dropped one of those when it was 21. So 18, let’s say. Several times, because I was very interested in so many different things. And I had straight As in high school, other than my first semester of ninth grade, and I had straight As in college and it was just because those things came easily to me.

Brock: Was there any particular angle that you were taking through your philosophy studies? Where you more on the logic side or on...

Rudkin: I took a symbolic logic course, very useful to-- which was a funny course because it was required of graduate students in philosophy to take this course. It was easy for me because it was basically mathematics. And very much recently, let’s say 6, 7 years ago, I met somebody who it turned out was in that course and he was a philosophy graduate student at the time, and he still remembers how-- it was easy for him too, really-- but he still remembers that we were in there together and some of what the instructor did was a little daunting. Let’s just put it that way, so. I took a course in ethics. I took a an entire course in Plato. I took a course on Dante, which was not, actually-- “Divine Comedy,” I should say, which was in the-- not in the philosophy department but it was taught in the Italian…I guess it was probably taught by the Italian department or something, but it was sort of a multi-disciplinary course really. So I loved reading “The Divine Comedy.” There’s just a lot of things that I found interesting.

Brock: It sounds like you were taking a very substantial course load. Did you have much opportunities to do anything else during those years? Did you have any other activities that you pursued?

Rudkin: Well, let’s see. What did I do? I still bicycled, not a lot but some. I had a bike in Bordeaux, that I bought that-- it was very good for getting from where I was living to campus and out in the countryside and so forth. Boy, I just don’t remember. I’m sure there are but I can’t think of them. I mean, I had some things that-- for example, I was on a committee, if you will, as a student member, that ran a couple of the scholarship programs. The scholarship that I had and some others that were combined into this one committee and so we had a lot of activities and we also interviewed the high school seniors who were coming in who were potential candidates for the same <inaudible> scholarships, et cetera, et cetera. But really I don’t really remember-- a lotta basketball, Kansas basketball. When I was a sophomore, I went to the Final Four and so my friends and I went down to Houston and inside the Astrodome to watch that. They didn’t do well there but they did very well on the season as a whole. Lost in the semi-finals to UCLA, but then everybody was losing to UCLA that year. But I wasn’t really active-- I have a fast heartbeat that kept me outta the draft. That was a good thing but it also meant that I couldn’t do a lot of activity after about ninth grade. So some of the things that I might have wanted to do, perhaps like playing basketball, pick-up basketball or something, I really wasn’t doing.

Brock: When was it that you first had an encounter with computers to computing?

Rudkin: Ah, yeah, so unlike Dennis, I had no opportunity to see a computer until I got to university. We did have-- take that back. Boeing Aircraft is there in Wichita and they had an open house one day and-- when I was in late grade school or junior high and they had-- they were showing this IBM punch card sorting machine, and then they gave us a punch card afterwards. And I thought, “Well, this is cool.” So that’s the closest I had ever come to an actual computer at that time.

But my freshman year, one of the courses I took very first semester was Computer Science 16. They were numbered starting at 1 instead of 100s or whatever they do nowadays. CS16, which was introduction to programing, and we learned a little bit of several languages. Learned FORTRAN, learned Algol, learned Snowball, learned some others I don’t remember now. Just enough to know what programing was. It wasn’t dumbed down or anything like that. I mean, you had to know what you were doing to write the programs and get them to work the way-- to solve whatever problem you had been given. But that was actually the only computer course I took as an undergraduate ‘cause I wasn’t even thinking about computers as a career. But I did get a job. I had jobs on campus, part-time jobs on campus. One of them was in a high-temperature chemistry lab with a distinguished professor, Paul Gillis, as the head of the lab, and he would or other people would have me do-- some things where chemistry related, you might say. Some were typing grad students’ thesis or dissertations because there were no computers to do them on so they always had to be typed. And so I’d be given a markup of a previous draft and I would type it again within the-- I was a very good typist from having taken it in ninth grade and I thought this was not a very hard problem for me.

But one of the jobs I had was doing programing on the university computer, which was a General Electric 635 or 645. I don’t remember which. There was this large program for doing least squares analysis and it was written in FORTRAN and my job was to make bug fixes or enhancements that some of the post docs or the grad students needed to be done. And I had that job off and on throughout my four year-- well, my three years in Lawrence. Probably four of my eight semesters I had that job. So that was my programming introduction beyond just taking that one course. Sometimes we di-- that particular project I was able to do on a teletype, sitting in the chem lab or the offices or rooms in the chem area. We’d sit down at a teletype. All uppercase, you’d type it on punch tape then you’d take a punch tape and read it in and it would go over to the computer center all at once because you weren’t really-- you weren’t running a time-share system really. You were just using it as an input terminal that was faster than punch cards, let’s say and easier to correct ‘cause you could see it on the sc-- well, there was no screen. It was paper but least you could print it on the paper to see it if you needed to.

Also, my second semester, I had a physics course, my first physics course ever, in which we had an assignment on the computer. But we had to learn basic. This was Dartmouth BASIC, which was-- I think it had only come out about seven, eight years earlier, and that was another one where we sat at a terminal and created the program. And it was a simulation of, I don’t know a lunar landing or something like that, where you had to take into account the mass of the thing as you’re burning fuel and the gravity of the moon as you’re approaching and all these other things in equations that you would put in BASIC, in the program. And generally speaking, you-- and then you would do, like, a simulation that would say, “Okay”-- it was like each time interval, “What has changed since the last one?” Well, you’re this much closer.

You’ve lost this much mass but you’ve also gained this much velocity. And it turns out that near the end you really needed to be doing intervals that were like milliseconds <inaudible> in the simulation, else you would-- the changes in one second were so massive that you couldn’t really model it in at this granularity. So I learned a lot about that kinda problem too doing that project.

Brock: What had motivated you to sign up for Computer Science 16?

Rudkin: Well, when I was-- so I had a friend from high school. He and I had both done debate together and we’d done a lot of other things together and we actually even moved our stuff from Wichita to Lawrence together. We didn’t live together ‘cause he was living in a off-campus and I was living in a scholarship hall but he was signed up as an engineering physics major. He, like I, was very good at math but had taken chemistry and physics-- well, I’d taken chemistry in high school too but he had taken physics in high school also and he decided he wanted to be an engineering physics major. And so he said, “You know, I’m gonna take this computer course,” and I said, “Hm, that looks interesting. Let’s take it together.” So we did. That was the reason.

Brock: Did you feel any particular affinity for it or did you...

Rudkin: Well, I thought-- I don’t remember really but I knew that it was interesting. It was not very difficult really. It played into a couple of my strengths, both logical thinking but also a capacity for tedium. You have to get it exactly right or it’s not gonna work and since-- unlike my job and this physics class I was talking about, which we did at a terminal-- albeit a very primitive terminal, no screen just paper teletypes-- unlike that the course work was always done by going into the key punch room, putting your program on punch cards, taking it to the desk or the window to submit it, and some period of time later getting back your cards and the printed output, the output including both the program listing and also the results of whatever you did. And rarely was it right the first time, of course, maybe the first <inaudible> assignment which was dead simple like teaching us how to put the right set of cards together, the right control cards and the right et cetera, et cetera. That was pretty easy. But, after that, you always had some mistakes. So this was a turnaround that you had to worry about. It might be an hour later. It might be four hours later before you got the job back. So you would be very careful with your time, with getting it right, before you submitted it. You would do sorta these mental simulations. “Well, what happens here? What happens here? What happens if this condition comes in? What have I forgotten about?” And these were simple programs obviously but, in your first class, they have to be simple and you still have to learn how to-- it’s not totally daunting but it has some tedium to it.

Brock: You were saying that you had a particular tolerance for that or...

Rudkin: Yes.

Brock: ...capacity for it?

Rudkin: Yes.

Brock: Yeah, that was...

Rudkin: Attention to detail is another way probably of saying the same thing, and almost a compulsive attention to detail. The kinda thing you have to do in that field.

Brock: Well, as you near the end of your studies, what were you thinking about for yourself in terms of a possible career path and a possible next step from the university?

Rudkin: Yeah, so I didn’t really know what I wanted to do. I should point out here that as a sophomore, February of my sophomore year, I met my wife. She and I both lived in scholarship halls which were small units, 50 men or 50 women. We did the work of the house. We did the cleaning or cooking or whatever. Everybody had a job for a room and board that was about two-thirds of what it was in the dorms. So that was the benefit of living there. But being small, they also were fairly close-knit between the people living in one and also between the halls. So a guy who had been my roommate the pre-- there was four to a room studying and then a sleeping dorm upstairs. So, when I say roommate, I mean, really, study mate. He had been one of my three roommates the previous semester, the fall semester, and his girlfriend, who lived in another hall-- my future wife, was one of her roommates and that’s how we met, playing bridge. They needed a fourth and they got me.

So, by the time I went away to France, we knew we were gonna be together but this was such an opportunity and she really couldn’t go so I was separate from her for that nine-- well, it was really more than nine months because I stayed on another month in England. But, during our senior year, we were planning to get married when school was out. We got married the Saturday after graduation ceremonies. We applied to graduate schools together. She had a degree in anthropology and chemistry. She wanted to pursue anthropology at graduate school. I was accepted at Illinois with a really good offer but she wasn’t so that ruled that one out. We were both accepted to Wisconsin. I was applying in math, you see, and she in anthropology. We were both accepted there and we both got some support, and so that’s where we went in the fall of ‘73.

But before that I considered a number of things. “Do I want to be a mathematician? Well, that sounds kind of interesting but what else do I might wanna do?” And one of the things I considered seriously, that I got to be very interested in was the history and philosophy of science. And I looked into programs like the one in Berkeley that I thought but ultimately, I guess I would have to say, I was a little bit risk averse by this time, unlike when I was a freshmen. I could do anything. Now I was gonna be married and we were gonna have to worry about supporting ourselves because we probably weren’t gonna be getting enough money from graduate schools to completely live on. So I sorta became a little more risk averse and a little more concentrating on this type of career and livelihood and so forth.

So we went to Wisconsin, went to Madison, moved there in ‘73. We were there ultimately for three years but in the summer after our first year of study, I took a computer science class ‘cause I decided that would be my minor for my math PhD. I took a course over the summer in assembly language programing ‘cause that’s kinda-- I was really having to start from scratch almost. I had programed but not much. And that was a pretty good course. And then, in the fall, I took another one and in the spring I took another one. These were like taking one computer science course while taking, I don’t know, two or three math courses, whatever it turned out to be. By that time, by the way, my math-- I had decided my real interest was in logic, set theory, foundations, that field. I guess, broadly speaking, it’s called logic but it really has

all these elements to it. And there were some very good math professors in that part of the math department. Jerome Keisler had written a book called “Model Theory,” he and a Professor [C. C.] Chang, who I don’t know where he was from. But Keisler was one of the professors I had for model theory. I think it’s safe to say they wrote the book on model theory. Ken Kunen-- I took a course on foundation-- not foundations but recursive functions theory, which turns out is very useful in computer science also. And, as a senior in college, I had taken an independent reading with this philosophy professor I was telling you about -- that I took symbolic logic from. Now, I’m a senior, I’d taken readings in a book called “Metamathematics” by Stephen Kleene, and that’s basically this whole field of the logic of mathematics. It’s more than just logic. I mean, it’s proof theory. It’s Gödel theorems, et cetera, et cetera, et cetera. And that book was written by Stephen Kleene who was also a professor at Wisconsin, although he was not teaching any longer because he was fairly old. He, it turned out, was one of the founding members of the Computer Sciences department at Wisconsin but I didn’t know that at this point.

So after this second full year of graduate school, I decided, “I don’t really think I wanna be a mathematician. I don’t know what I would do as a mathematician. I don’t really wanna be a professor of mathematics.” As an undergraduate, I’d taken so many things and I was top of my class and so forth and so on. There were people that I met at Madison who had done nothing but math and related fields and were way better than I was at math. This was the first time in my life probably I wasn’t at the top of the classes that I was taking-- some classes, but many not. So I said, “Okay, I’m gonna switch from math to computer science,” and I did that officially going into my third year in Madison, by which point I had taken, I don’t know, five or six courses over the previous five semesters, whatever it was, including summers, two summers. I had a teaching assistantship in the Math Department my second year and then instead I had a teaching assistantship in the Computer Sciences Department my third year. So, that’s how I got to be in Computer Sciences, and by the time I was in my third year, third and final year: compiler course, several really high level operating systems courses -- artificial intelligence was another field I took several courses in throughout those years. Frist class professors, learned amazing things about, not just about computers, but about programming as both an art, but more importantly as a logical enterprise that requires careful attention to so many different things, attention to detail, yes, but also structuring your thoughts on your programs to come to the results you want. I was introduced to Edsger Dijkstra during this period and read his “Structure of Computer Programs,” I believe was the name, something close to that, about that time. Actually, I think I might have been introduced to some of his papers independent of the book while I was at Madison. I don’t remember the chronology there. The compiler course from Charlie Fisher, who’s now Emeritus Professor, and who I’ve seen recently because I’m still involved with the Computer Sciences Department at Wisconsin. He taught the compiler courses and really taught us a lot about not just-- well about programming languages and how they’re built and how to evaluate a good feature from a bad feature, the language, what kinds of things make it harder or less hard to compile. And, also, before you get to that the languages themselves have a syntax and how do you evaluate the syntax and learning about different syntactic models. Is it LL or LR, left to right, how you deal with the way in which it can be parsed. All these things were just very, very interesting to me, both from a computer perspective, but also from a logic perspective.

Brock: It seems that there’s such an overlap between those domains and even in some of the same characters I would imagine you know from...

Rudkin: I think there might be. The origins of the syntactic, the syntax representations, usually you use what’s called Backus-Naur Form, which was invented by John Backus, and I don’t remember Mr. Naur’s first name, in the `50s possibly when they were doing Fortran, I guess I don’t know that for sure. But, it is not very different from what Chomsky and other people did with natural language syntax representations. I’m just now talking about at the level of writing the syntax, not so much as how you parse it. So, I think there was a lot of crossbreeding there between the linguists, I mean I know there was, between linguistics and computer science. And, I’m interested in linguistics now too. I never took a formal course in it, but more recently read a lot of books on language. Linguistic books as well being books that are general about language and specific about say Anglo-Saxon or other European languages.

Brock: I would imagine that just in the same way that there are many, you know, once you were at Madison there were many directions you could go within mathematics and you were choosing this more like foundations of mathematics, logic area, I would, perhaps I’m wrong, but I would imagine that you had a similar sort of choice in where you wanted to go within Computer Science from -- I was wondering if you could talk about if that was true and how you zeroed in on kind of what you did zero in on?

Rudkin: Well, I consider myself a compiler guy. I had done a number of other things, however, that were just as interesting almost. In the artificial intelligence area, there was a Professor named Larry Travis who I took some AI classes from, who had a project that was partly-- he had a grad student who together with him, Larry, were working on a language they called TELOS, T-E-L-O-S, and I don’t remember why they called it that exactly, and they needed a simulator actually. It wasn’t really that they were going compile it, but they needed to sort of test this language out. This had a lot of AI features in it was the reason they were doing it. Of course, that was a very common thing that people were doing in those days, “What kinds of languages should you use for doing artificial intelligence?” Okay, some people started with LISP and did this, that, and the other, and there was a LISP variant at Wisconsin that a previous grad student had done that was very, very advanced in the sense of things they had added to it that made it easier syntactically and in other ways. But, the project I worked on then, this turned out to be my last semester at Madison, was to do an interpreter for this language so that I could syntactically analyze a program in this language, and then run the program on a simulator. So, we weren’t going to machine code, but we were just sort of running as a simulation execution. And, I wrote that in a language-- it was a compiler and interpreter in a language called Simula. Simula came from Sweden and it was called Simula 67, in the `60s, but a group of computer people, I don’t unfortunately remember their names anymore, who needed a simulation program, and they had decided to take ALGOL, the sort of standard that most languages had ultimately derived from in those days. ALGOL -- and Pascal was really based on ALGOL and so forth. They made a language that added object-oriented featured classes, instances of classes, creating new instances and then having functions that you could export from a class that you would then be able to call to do something. And, if you made a sub-class, an inherited class then you could have the same function that’s the same name, but depending on which sub-class the object actually belongs to something different would happen. All these features were in Simula and they built this language as I understand it in order to simulate things. They had a lot of features in the language that allowed you to do simulations of various kinds, like traffic flows or something like that where you have to model the cars and the other things, but you also have to put some variability into it so that the different kinds of arrival times onto the scene and what would happen if you run this way and that way and the other and the results. They had a

lot of features for that, which I was not interested in. But, it was an object-oriented programming language, and to me, first of all that was, I thought, a very good model for programming: encapsulating data in such a way that you hide the details from the outside world. You have an interface that’s public, but you have an implementation that’s private. This has become common and C++ is based on this object concept. Java is based on this. There’re a lot of languages that are, but this is the first time I had ever run into it. So, again, what I was doing was essentially programming language implementation in this project I was doing for the class.

When it came time to interview after I finished, you know I knew I was going to finish in May of `76, I was interviewing that spring. That was what I looked for was that type of job. One of the places I interviewed was at Burroughs, near Philadelphia, Tredyffrin, I believe it was called, where they were doing large Burroughs computers, I couldn’t tell you anymore, B7 something, I don’t really remember any more.

In all my classes, Burroughs was always touted as the gold standard in a design of architecture that basically as far as I could tell invented virtual memory, or at least used it very early. The machine language was basically ALGOL. It wasn’t a machine language, a number of features like this that just to me seemed the right thing to do to be able to take a lot of the error in programming and greatly reduce the kinds of mistakes that you would make. I mean that’s really the ultimate goal, I think, of a programmer, or the person that creates the tools for a programmer actually, put it that way, is to make it easier to do a program that works than one that doesn’t work. And, this point, by the way, was driven home to me many years later when I was working at VisiCorp and I was programming in C and there were some people who had done a lot of C programming and this was the original C. They hadn’t added some of the type checking that they did later. One of my friends said, “You know I’ve been doing this a long time, but I just realized how easy it is to write mistakes and how hard it is to find them in C.” And, I think that’s an important consideration for a language designer and for choosing a language to use when you’re doing a project. Anyway, Burroughs, but the fellow with whom I had written my compiler project in my compiler course, he and I went to Burroughs together, we interviewed together. We both got offers there. He ultimately took that and stayed in the East for many, many years. I, on the other hand, also interviewed at Intel. There was an ad on the back of “Communications of the ACM” that ran for many months, if not years, that said, “Do you speak our language?” And, then they had a bunch of words like pushdown stack and I don’t know a bunch of words there, and “If you do, then contact us at this address or phone number.” It turned out they were not recruiting out of school with this ad, but I applied to it and they had me come in. I mean, in other words, they had a separate technique for recruiting directly out of school.

Brock: Okay.

Rudkin: But, I got the interview, I went to Santa Clara, interviewed with a group that was doing the software for the development systems that supported the 8080 and other microprocessor chips. And, I was made an offer and Jan and I had to go are we going to East or are we going to West? I’d lived in the East, well not very much, I lived in Ohio. I’d lived in the Midwest. I’d never been to California before this interview trip. I’d never even been more than a few miles west of the Continental Divide, so let’s go west, so that’s what we did. But, even that job was not a compiler job, but it had some kind of compiler elements to it, or language elements to it. I guess what I’m trying to get to here is I thought this was the field I wanted to be in, but it didn’t really matter because there’re so many interesting things in computers

then and now, and you don’t really have to know a whole lot about it when you start because ultimately if you’re smart enough you can figure that out. It’s the computer science part that’s crucial, not the particular domain of computer science. Now, if I were going to do something in artificial intelligence maybe that would be different, I’d probably have to know a lot more about that, and I’ve not kept up with that field today. But, I did a number of very different things over my career, none of which I would have anticipated as an undergraduate, as a grad student either. And, I think that’s key to an education. You shouldn’t be trained to just know one thing or -- some schools seem to think they’re trade schools. Why didn’t you teach me Python, why didn’t you teach me FORTRAN instead? Well, if you were smart you’d know how to pick up the second one after you knew the first one. You know, you just have to know how to learn, right. You have to know. I won’t beat that horse any longer, but that’s why I believe in a liberal arts education. That’s why I have a BA and not perhaps a BS or an engineering degree.

Brock: In thinking about while you’re in Wisconsin and you describe yourself as I suppose then you would’ve described yourself too, as a compiler guy...

Rudkin: Yeah, that’s my most knowledgeable field.

Brock: Okay, what I was curious about was if there was then kind of like a status hierarchy within computer science of being interested in compiler versus something else? Was there any of that?

Rudkin: Probably, I don’t remember. I would say this. The hierarchy might be more like systems level people, by which I mean operating systems and languages and of course operating system that covers a whole lot of things. But, those skills and so forth are probably perceived to be harder and more prestigious than applications. Now, things have changed a lot in the intervening 40 years as to what kinds of things people need to work on. There aren’t that many new computers in the sense of a brand new architecture. There are a lot of things that people have to do to keep whatever architecture there is still running to make enhancements and fix problems, et cetera, et cetera, but it’s always been the case I suppose that the bulk of programming is in applications. If you go back to when I was in school, one of the things I learned about was what they called the software crisis, and this was something that magazines like “Computer World” were touting as this huge problem with not enough people who knew how to write good code. And, there were so many applications that weren’t really working well enough or it wouldn’t be able to be maintained or whatever, and when I heard that I didn’t really know what they meant by applications. I eventually figured it out that in that context an application was not just a program, it was something you were doing in your business to run a certain type of report. It might’ve been a database related thing. It might’ve been accounting. it might’ve been accounts receivable or what have you. It might’ve been inventories or factory flow of, you know, whatever your business was you had these kinds of things that somebody had to write and there were a number of tools that people had created to make those applications easier to do and a number of people in each company that did that kind of thing for the company. And, then IBM did this as a big business. To me, at that time, I thought that field of computers is okay let them do it, but I’m not interested it’s not very challenging. It doesn’t take much brain. It takes maybe a lot of work, but it’s not really an intellectual exercise. I don’t know if that’s true, but that was the way I felt about it. When I was growing up we used to refer to this type of job as matchbook programmers because when I was growing up you had these ads in the matchbook covers for a school to learn how to, and probably they were teaching COBOL or something like that, I really don’t know. To me,

that was a derisive term to call them matchbook programmers. I’m sure their skills were definitely needed, so I don’t mean to denigrate them, but it was not anything I was interested in. But, what I thought was an application would be something more like a text editor for some other kind of program that ran on the operating system, but wasn’t the operating system, and those were important too, more important to me than these others. But, they were still not as interesting as systems level. But, you know, this whole spectrum from way down in the bootstrap code or whatever else you’d have at the bottom, up to applications, it’s really a continuum and it always has been, and I have a friend who wrote a paper called something like “Concept of Operating System, Disproved,” not disproved, but anyway his point was well what exactly do you mean? Some people include all the compilers as the operating system. And, nowadays you say, “Is the browser part of the operating system?” Well, not really other than in a marketing sense or a legal sense, but really it’s not. The operating system is the really crucial stuff that everything depends on.

Brock: Well, if we return to -- you come to Santa Clara to the Intel Headquarters, it’s 1976, this means that commercial microprocessors are only on the order of six years old or so...

Rudkin: Oh, less than that `72.

Brock: ...yeah...

Rudkin: The 4004 came out in `72.

Brock: That’s right.

Rudkin: That’s when it was commercially available.

Brock: Yeah. So, very early in microprocessors. Could you just talk about what you thought about them, microprocessors as an embodiment of computing and what you thought of the software opportunities around them? Just what you thought about it.

Rudkin: Well let’s see, of course, they were very primitive as processors relative even to minicomputers or mainframes of the day. They had much smaller memory, they were much less rich instruction sets. But, and I think it’s probably safe to say, that the primary usage in that period was not in computers per se, although of course they were some computers were built with chips at that time, I mean the Apple II. I forget now which chip it had in it, but it wasn’t Intel’s, but I can’t remember now who some of the other manufacturers were. The Z80, but it wasn’t a Z80 either. Anyway, the point is that there were some-- and there was the IMSAI and the other things that Bill Gates and Paul Allen were building software for, well building computers and software for, you could do those things, but what I’m getting at is that the vast majority of the count of processors that were manufactured were in embedded systems of one kind or another: traffic controllers or things in a factory floor or something. And, you know I didn’t really care that much one way or the other what the uses were, I just knew that my job was, it turns out my job during essentially the whole time I was at Intel was working on debuggers. There was a particular kind of debugger that worked for debugging both hardware and software called an ICE, an in circuit emulator, and my initial job was to make some changes to the ICE 80. So, by this time the 8080 was out and had

been for a while. The 4004, 4040, is there a 4040, I’m not sure, but the 8008, those were no longer the big dogs. There were some others also, which I don’t remember now. I don’t remember the numbers at all, but the 8080 was the power horse, you know the main work engine for this business, and there was a development system called the Intelec MDS, standing for Microcomputer Development System, and the group I was in at Intel, in the Microcomputer Division, later became Microcomputer Systems Division, was doing the software for these development systems, that included compilers for a language called PL/M, Programming Language for Micros, I suppose was what it stood for. There was a very simple operating system that did the things you had to do. You know, it did disc management, not management, it made sure that the terminals and the screens and the disks, you know that everything could boot up and run and they shipped on 8 inch floppies that were, if I had to remember I’d say 2,000 bytes, I don’t know something amazingly small. They were just tiny, little things. They had to be bigger than that, but they weren’t very large. The whole machine only had either 16 or 32 kilobytes of random access memory. Our program had to run-- the operating system took 12 kilobytes. The program that we ran, like for example the debugger, had to be no more than an additional 12. This doesn’t add up. The point is that you had to leave room for whatever the user was doing on top of the operating system and the debugger.

Brock: Right.

Rudkin: I may be off somewhere, but 12K is the number that sticks in my mind. So, one of the things I had to do in that first-- the reason that I had this task at all was to adapt the ICE 80 that already existed for a language that was compiled on a mainframe to a new version of the MDS operating system where everything was local on the machine, the compiler would actually run on the machine.

Brock: I see.

Rudkin: So, I had to make some changes because this was a little different in one way or another. But, the big change was because of size and things that keep growing as software always does, it had to have overlays. It had to have a piece that you would load in the memory when you needed these functions and then if you needed these functions you had to load this in, which would replace this. So, you had to very carefully figure out which things you want to be in memory all the time, and which things are you-- I mean it’s basically a segmented memory type in a very rough sort of way, except it was all done manually. There was no operating system support except the load coming up. You had to figure out when you needed something in that module and load it before you called that function. And, then you had to figure out that now you need something in a different one therefore you’re going to have to give up this one, and let it be overwritten. This was about a five, six month project. It wasn’t a big project, but it got me to learn the whole concepts of the in circuit emulator of this ICE operating system, which interestingly was called Isis. The fellow, I think, who named it had a big interest in Egyptian things. The compiler was coming, the locally hosted compiler. I don’t think Intel was building it. I think somebody else was building it for them, but I could be wrong about that. But, anyway these things were all coming together and we had to make them all work together. And, after that project finished then we started working on the in circuit emulator for the 8085, and then a year later the 8086. And, I was in charge of the 8085 project, the software side. Now, these ICE also had hardware because it wasn’t just for debugging software it was for debugging the hardware design as well, and you had an umbilical cord that went from the machine you were running the debugger on to the hardware you were debugging. Took the 8080 out of that hardware and plugged what

is the same form factor in a 40-pin DIP that you had on your umbilical cable into that same socket and now you were controlling from machine one, you were controlling what was happening in machine two, both hardware and software and you could set breakpoints, which would mean if you ever execute this address of that processor then stop and give me control back so I can see what’s happening now. And, doing this in hardware meant there was no-- I mean you can simulate a program and set all sorts of debugging things, but it’s not like running the program at full speed, which this allowed you to do. And, then when it stops, when you get some condition, which was usually an address that you’re executing, but it could’ve been reading from a certain piece of memory or writing to a certain piece of memory. Whenever one of those conditions happened control would come to the debugger, you could do whatever you needed to do. And, so, when that project began, we decided okay we need one for the 8085, we will need one for the 8086. There was another processor, I think it was called the 8047, which was not as powerful and it had some different target market, I guess, I don’t really know what that was. And, there were two other projects that came to exist, but they all were roughly speaking debuggers or there’s another name we used besides just debugger because the whole class was more than debuggers, but it was hardware development aids. Let’s call it that for a moment, and we had a hardware architecture for support so that we could write-- so that the folks who were doing the code that lived in the pieces of hardware basically went into ROMs or PROMs or something.

Brock: Yes.

Rudkin: And, could kind of share code between these to a certain extent. They had an interface between the hardware and the debugging machine that was standard. You know, it would put things in a little piece of memory and then you would send that piece of memory, that little structure off to there, and those were all kind of standardized and then the software itself had a common architecture where there were a lot of shared pieces to it that all these debuggers could use and then each debugger had its own things that were. And, it had a command language that was a little different from what they’d had before because we added some loop, you could write basically scripts, so you could say something like “Start this up and the third time it hits this address then give me control.” And, the hardware just said “Okay I hit it,” and the software would say “That’s only one.” “Okay, I’ve hit it.” “Okay, that’s two.” You know, in other words you were implementing partly in this hardware that was really running everything down at the, I hate to say the machine level, but more or less, and partly then the software, the debugger, would take over do something, but wouldn’t necessarily bother the user until these conditions were all met. And, the language was something that I was the chief designer of, so I finally got back into language is what I’m trying to say.

Brock: Okay, now the language for?

Rudkin: The language for the scripting and the commands for the debuggers.

Brock: I get it.

Rudkin: Now, this is a very thin-- these were command languages, you had a prompt, you typed something. This was not even remotely graphical, not even remotely. But, it did have to have a scripting language, a command language that included scripts is a better way to say it.

Brock: Well, it’s interesting to me that it sounds like on these development systems you have a piece of software that’s an in circuit emulator, if you will, so this allows the user of a microprocessor or the customer of a microprocessor to develop the program that the microprocessor will execute...

Rudkin: That’s right.

Brock: ...and perform some function in some piece of equipment.

Rudkin: Some piece of equipment.

Brock: And, then it sounds like the next step is to actually be able to have the development system basically play the part of the microprocessor plus the ROM in the actual equipment.

Rudkin: Yeah, that’s right. That’s pretty much it. I mean you are emulating, you’re not simulating. That’s why it’s called an in circuit emulator.

Brock: Emulator.

Rudkin: You could simulate a program by running something that slowly steps through it. Usually it’s slow, not always, and does whatever you want between steps. This is actually running on the target processor 8085, let’s say, in an environment that included other components on a breadboard. You can have conditions where if a certain register is written to by, let’s say, there’s an input device that’s looking at a signal from a loop in the pavement for a traffic controller, you could simulate that signal being sent and what it does in the processor and then what the hardware should do with it and also what the software in the target machine should do with it if that’s what you needed to know about. That’s the level of thing you could do. You have to remember these processors, this was an 8-bit bus, 16-bit registers, but only a few, I don’t remember certainly less than 16 registers. There was a stack pointer and a program counter, and a few maybe eight total registers I would bet. I don’t really know for sure anymore. I mean these were by comparison to even 8086, which was the first 16-bit processor that Intel did, these were very primitive by comparison.

Brock: Well, it also sounds that in this experience that you’re having at Intel, the development system is a personal computer in the sense of it has all of the--

Rudkin: That’s right.

Brock: It’s not framed in that fashion. It’s framed as kind of an engineer’s tool to do certain things. But as you describe it, it has-- it’s very interesting because you’re having this experience there of it’s a small machine, microprocessor based. There’s an operating system. There’s a--

Rudkin: Yeah. I think it was sixteen kilobytes of RAM was the minimum. And the operating system plus our debugger used twelve at most. That’s the kind of constraint we were under.

Brock: Right.

Rudkin: Yeah, it had a cathode ray tube, a monitor.

Brock: Right.

Rudkin: Attached to the box that had the processor, and the memory, and the disk if there was-- I mean, there was floppy drives, one or two depending on. There must have been a hard drive, but I can’t really remember for sure. The terminal had the keyboard. There was no concept of a mouse in this domain. But still what you had was a computer, yeah.

Brock: Yeah.

Rudkin: It just couldn’t do very much. It was not very fast. It didn’t have much memory. It didn’t have much-- the floppies were, I really want to say two thousand and two or four thousand and four bytes. But I maybe off somehow. Those numbers just stick in my mind. That suggests you couldn’t even put the program on one. So, that’s probably not right. I’m not sure. I’m not sure.

Brock: Okay.

Rudkin: And they were the eight inch floppies. But this was before-- over time, many things have got better exponentially. The amount of data you can put in per area has gone up incredibly. And that’s one of those things as well as processor speed and so forth.

Brock: So, you were with Intel until 1979, is that right?

Rudkin: Yes. I didn’t stay terribly long. I had a good time, but I heard the siren song, and I went somewhere different. That didn’t work out very well, so then I went to Bell Northern Research.

Brock: The siren song was to a different company.

Rudkin: A different company, yeah, that I knew some people that had gone there before me. But when I got there it really wasn’t--

Brock: What you thought it would be.

Rudkin: Well, it was another chip company, and it was not-- they were not a particularly good chip company. But if they were doing this project well, that would have been worthwhile. But there were some structural things in the company that kept that from really happening. So, yeah.

Brock: Was that Zylog who would have been?

Rudkin: No.

Brock: Would you like me to guess?

Rudkin: No. No, I’m not going to say.

Brock: We’ll skip that. I tried. At least you can count that I tried. Well, so there briefly. Well, then how did the connection come to Bell Northern Research? And maybe you could talk about getting there and what you started to do.

Rudkin: Right. So, my connection was the fellow who had worked with me this interim six months. And then he had left before I did and went to BNR. He had some friends who worked there from his Burroughs days down in Santa Barbara. He knew Dennis Austin very well from Burroughs days.

Brock: Okay.

Rudkin: But he had some other friends who worked at BNR. And he went there into the computer science research group, which was at the time fairly small. And Bob Gaskins was the leader of it. John Ahlstrom, the person I’m talking about who I followed, was one of them-- was there. The two big ones were Whit Diffie and Ralph Merkle, who had done public key encryption work with Marty Hellman at Stanford and had published that already. About a year before I got there I guess is when it-- no, sorry. Let me rephrase that. I saw it in Scientific American about a year before I got there. So, I don’t know when it was actually published. So, they were in this computer science group. And there were some other people. There was woman who I’d known at the Wisconsin CS department. Although, I didn’t remember her at first. There were maybe half a dozen of us working for Bob I would suppose.

And the group, computer science research-- the thing you have to understand about Bell Northern Research, it was the Bell Labs equivalent, if you will, to the Canadian phone company, Bell Canada and its Western Electric equivalent, Northern Telecom. And BNR was owned jointly by those two entities in Canada. And there was a BNR Inc. We were at BNR Inc. There was a Bell Northern Research Ltd. in Ottawa that we were connected too in the sense that our boss’ boss’ boss or something was up there. So, everything in the company at large-- I mean they were working on-- they had-- in our building, there was a group doing the PBX-- or sorry, not the PBX, the Telcos-- the switches and so forth for the Swedish phone company. So, I mean that was a big, real project where people were creating things that were in use. And there were some other projects like that. But our group was a research group. And we were looking at, broadly speaking, you might say that the technologies that one would need in the so-called office of the future, which was a big buzzword at this time. And what that really meant was being able to do things on a graphical user interface, high quality printers. And this is the goal, the office of the future goal, a lot less paper being used in offices, a lot of things integrated that weren’t. Now, much of this-- most of this-- all of this and more has happened. But in 1979, it was only in labs. It was at Xerox PARC, in Palo Alto, they were doing a lot of on this. Other companies were, too, that I’m not remembering and maybe never really knew much about. And it was just before I went to BNR, but through the people like John Ahlstrom that I knew that we went to visit PARC and saw-- because he had another friend from the old Burroughs, Santa Barbara days who was a scientist at PARC. And they gave us a-- this friend, Wayne Wilner gave us a tour of what they were working on, the Alto-- not Smalltalk perhaps, because he was not working on that, but all the other kinds of things that were being done on the Alto, the user interface, and so forth and so on. It was like, “Wow, this is really cool. I don’t exactly know where it’s going, but this is pretty cool. This is a lot better than typing commands into a prompt on a black and white, text-only screen.” So, my job at BNR was I was working on, again, a programming language. Well, was it really a programming language, or was it a user command language? But it was kind of like Smalltalk in very

rough forms. And I did a lot of-- a lot of what BNR did was writing proposals to get money and then writing reports, which I didn’t have to do. But a lot of what I did sort of supported in one way or another getting new funding from Bell Canada or Northern Telecom. But a lot of what I was doing was reading the papers, designing and implementing an interpreter for this language I was talking about. And eventually, after a couple of years when the Smalltalk 80 books were coming out, or just before they came out, I wrote and implemented an implementation of the Smalltalk byte code interpreter.

Smalltalk is an interpretive language. And it’s not compiled in the machine code, but it’s compiled into what we call byte codes very much like what Java is today in which-- so, these byte codes were run then by an interpreter at run time. And Smalltalk was an object oriented, but it also had a lot of graphical things in it. And it also it was, in some versions of Smalltalk, was intended for kids. Not this one that we were talking about here, but that’s what I think Alan Kay was-- Alan Kay’s Dynabook was, I don’t know, the iPad except it was thirty years earlier, forty years earlier, you know, the idea of sitting in the park and being able to anything on your computer. Yeah, how are you going to do that? Well, now we know how to do that. The concept was there, but the implementation wasn’t. Anyway, I wrote this interpreter for Smalltalk. And I think I wrote it in Pascal, but I’m not sure what I wrote it in. And I can just remember the first time I showed it to Bob. And I type in three plus four, and seven comes back. And there was an awful lot of stuff under the hood that worked to make that happen. And then I said, “Let’s try one other thing.” And so, I typed in a little function to implement factorial where it’s recursive. It says factorial of n is n times the factorial of n minus one. And so, you just keep going and going until-- so, I just typed it in, a function, the first time I ever tried any new function definition in this little implementation. And sure enough, it got the factorials correct when I gave it some-- the trial. So, that was pretty cool. But it had-- really none of this work had any likelihood of seeing the light of day because Northern Telecom could never be convinced that any of this was worth pursuing, first of all. They were a telco company, telephone company, at that time. They don’t exist anymore much because they didn’t have this kind of foresight. But it was fun to do, but it had no real world implications. And that was a little disappointing to me because I built products before that.

Brock: Right. And around you were the other members of the computer science research group doing similar kind of exploratory work?

Rudkin: Yeah, actually there was some very good real work going on, I mean by that, work that people saw. For example, a couple people were working on typography. And we knew that laser printers and other really high quality printers were going to have to be part of this whole process of automating things. And so, we bought a Canon laser printer, one of the first I guess. We had it in our area. We wrote some things. I say we. I wasn’t really doing much of this. There was an implementation of Knuth’s TeX typesetting program. I don’t remember now how we did this, what machine it ran on. We had some VAXs running BSD UNIX. And we had some-- no, not BSD UNIX. Actually, it was a different UNIX. We had some VAXs, and we had DEC 20s running TOPS-20. This TeX probably ran on the TOPS-20 if I had to guess. But I don’t really know that that’s true. And that did real output. And people throughout the whole building, the whole company, would come to see and to get their stuff printed in this fashion that was very much better than any line printer, or dot matrix printer, or anything else, which were the alternatives at the time. So, that was real.

Brock: Right.

Rudkin: A lot of it had interesting spin-offs probably but not while I was there. So, I’m not really aware of them. And then they closed that group down when Bob left about early ’84, mid ’84.

Brock: But you had left--

Rudkin: No, let me think about this. No, I’m wrong. They closed CSR down by about summer of ’82.

Brock: Okay.

Rudkin: And then Bob had a different job. He was doing things with Nokia in Finland to get Northern Telecom and Nokia working on things. And I don’t know what else he did. And he left eventually and went to Forethought in ’84. But I left about six months after they shut down CSR. I went to VisiCorp.

Brock: Right.

Rudkin: And worked on VisiOn for almost four years.

Brock: Well, I’d really like to talk about that for a time. But just to-- your computer science research group at Bell Northern, that laboratory was in Palo Alto, is that correct?

Rudkin: It was in in Palo Alto on Porter Drive when I joined. They were building a new building in Mountain View. But before that our group, they ran a space in that building. They took over part of a building at Page Mill and Hanover, which was also ADP, I think. That building I just noticed has been torn down very recently. I mean in the last year, it’s been torn down and something else rebuilt there. But we were there for six months after I joined on Porter Drive, then another six months on Hanover. And then in the summer of 1980, say July maybe of 1980, had the grand opening at the building in Mountain View, which was at the corner of 237 and Middlefield. 237 was not a freeway then at that point. It was a surface street. But there was a traffic light there at that corner. And that was a new building. And half the building was us. And half the building was GTE, some group within GTE, which doesn’t even exist anymore under that name. I guess they became Verizon. I’m not entirely sure of the history there. But we had about half the building, which was well separated. I mean it was like two buildings almost that you could not go from one part to the other. And that was a wonderful building. It had great views, great ambiance.

Brock: And where was-- could you describe how you made the transition to VisiCorp and where that was located?

Rudkin: Yeah, so after CSR was shut down, I was working in another group. And it didn’t really seem that it would be going anywhere that I was interested in. I can’t even tell you now what it was. I think I went to Ottawa or Toronto once or twice during that period. But I can’t remember what I was doing. That’s how memorable it was to me. So, I start interviewing again. And I had known several people at Intel who had gone to VisiCorp, including the president of VisiCorp, Terry Optendyk, who had been the director, the second level manager of this whole software group at Intel. And he was the president of VisiCorp. And there were quite a few other people there that I had worked with before.

So, after first talking to them and then I had to sign a non-disclosure to see what they were doing, it was a project code named Quasar. And it’s what became VisiOn when it was announced. And it was basically take an IBM PC-- now, at the time about a year after the IBM PC was first introduced. But they had been working on it for most of that year probably, I would guess. At the time VisiCorp, which had previously been known as Personal Software, was publishing certain software. It was publishing VisiCalc.

Brock: Right.

Rudkin: There was one called VisiWord that came somewhere along the line. There was VisiPlot and VisiTrend, which I think was combined into PlotTrend, which was Mitch Kapor’s project. There was VisiDex, kind of a Rolodex. All of those, or at least most of those like VisiCalc, started on an Apple II and made the Apple II a very popular machine because people could now, in their department’s budget, purchase a computer and a spreadsheet that they could do their work without depending upon the central mainframe or minicomputer in their company. VisiCalc and Apple II were really successful. When I joined, these had already been moved to-- ported-- re-implemented on the IBM PC. And not a great implementation of VisiCalc because they didn’t really take advantage of the full keyboard. They kind of did a straight port where they didn’t-- there was a lot of keys on the keyboard of an IBM PC that you really would have wanted to use, which they didn’t support. I don’t remember the details, but I just remember that fact about the implementation. And I don’t remember if it got fixed or not. But the project I was working on, VisiOn, was totally homegrown within the company. So, it was not a publishing effort at all. And there was two guys, a company in Houston, called Rosetta. A couple of Rice graduates, Scott Warren and Dennis Abbe, very smart people who did the original architecture and not the UI-- well, part of the UI, but particularly I want to talk their architecture and then were implementing it on the IBM PC. And it was an IBM PC in graphical mode with windows. It was a Calc application, and it was running in its window. And there was a Word application that would run in its window. Now, none of those apps existed in much more than a demo form when I saw it. But the underlying stuff was all there. So, this was the thing.

And I joined in October of ’82. And we showed it at Comdex, November of ’82. And it was quite a sensation. And we all went-- well, most of us went down to Comdex to see it. But I’m told, at a time when I wasn’t personally present, so this is second or third hand, Bill Gates came by. One of his people said, “You know you’ve got to come see this.” And Bill Gates came by and said, “Aw, my people told me this couldn’t be done on an IBM PC.” That’s the quote I remember. I find out later, they were already working on what became Windows, but it was not public. And it was not very far along I think. And they really accelerated that. And about six months later, they announced Windows. We think to put fear, uncertainty, and doubt in us and the customers about us. I don’t know if that’s true. But, we thought we were going to ship in the summer of ’83. And it dragged on. It dragged on. And when we finally did in November of ’83, just before Comdex of ’83, the vice president of our group said, “Summer has finally arrived. We’re going to ship in the summer, and now it’s summer.” But it was November and that’s how much we slipped. It was way more work than we thought. And the problem was not the bugs but the speed. It just didn’t run fast enough. You can do these kind of small demo quality implementations, and they look lightning fast. But when you actually get down to building an application that really did the work that you needed to do, basically a VisiCalc except it’s in graphical, it was just too slow.

Brock: And so, the work would then be various means to try and keep the full functionality but somehow speed it up?

Rudkin: Yeah. A lot of the speed problems was from-- because of the IBM PCs in those days only had six hundred and forty K of memory, of RAM, and that was not enough really. You had to do a lot of swapping to and from disk of your program or your segments of code. And that was slow. That’s one of the problems. There were many problems.

Brock: And what was your role in the project?

Rudkin: Well, my role was-- I was the project leader on the systems level part, so exclude the applications. The two people in Houston, I was their liaison. And I was the one that was keeping them happy. But mostly, I was making sure that I knew what they were doing, and they knew what we wanted them to do and so forth and so on. And then I had three or four other people. I had one guy working for me who did the printing subsystem and another guy who did the file subs-- that is the being able to actually create files and show them in the Window, rename them, move them, copy them, all those things. I don’t remember much more than that. And then I did a lot of the user interface or the feature design on things that had not yet been designed. There was a feature for moving data between one app and another. We didn’t have a clipboard. We didn’t have cut and paste. I wish we did. We had something else called transfer, and it was kind of clunky. But that was my job to-- we had gotten to a point where we knew what we wanted to do, but we had to get the details nailed down because this app had to know what to do to talk to this app through this mechanism that I designed. You know, it’s kind of like-- looking back at it later, this was a very interesting goal. And I learned an awful lot about implementing systems like that and applications like that. And I also learned an awful lot about what not to do when, in particular, when your machine is not up to the tasks you’re trying to do.

Brock: And then after-- was it the case then after that the VisiCorp started to get into troubles right around the time that the product was actually introduced.

Rudkin: Yes. So, we introduced it in November. It was announced before that, but we shipped in November. It was shown at Comdex. We didn’t have a lot of sales. Moving into ’84-- and there were some other things going on. The VisiCalc people, VisiCorp had sued them because they weren’t delivering what we needed. And that suit didn’t go well. And the company was losing money. And it had lost the rights to VisiCalc. And I don’t remember what else. But basically in say May of ’84, the president, Terry Optendyk, was let go. And by July, a lot of the staff were let go. I mean some were let go over time as projects were canceled. But about from a height of two hundred and some employees, we got down to around fifty or less. And then they sold VisiOn to Control Data Corporation. Control Data in Campbell, this is the part of Control Data that was the old IBM service bureau that they had to divest of in the antitrust activity of the ‘60s. I actually don’t know when it was. And so, Control Data had this unit, whose name I no longer remember, mostly back in Connecticut. But we had an office here too. And the office here is where we all went then, those of us who were chosen and who accepted, which was probably, as I recall, about ten of us. And that was in summer/fall of ’84.

Brock: And Control Data was going to continue to produce the software and sell it?

Rudkin: Yes, but their product, in general, was the kind of thing that-- let’s see if I can think how this worked. There’s a name for it that I’m not remembering. The kind of thing where an airline would want to have pricing on its seats that would change with the number of seats that had already been sold and the number of days left until the flight.

Brock: Right.

Rudkin: That’s one application. And another was in a grocery store, the kinds of things they would do, which brands would they lower the price of because-- or maybe it’s the manufacturer, as opposed to the store. But by the metrics that they would measure and the software that they were producing would help you, the marketer of whatever product, know when it’s time to raise the price or lower the price in response to this, that or the other. That’s the type of thing they were doing. And I don’t remember any longer what they thought they could do with VisiOn. I don’t think their plan was to sell it as a product to the general public necessarily. But I don’t really remember anymore. So, we did another version or two of it as Control Data. But it was clear it wasn’t going to go anywhere. But it was kind of a nice place to work in many respects. So, I stayed until May of ’86. I went to work for Forethought May 1st of ’86.

Brock: And did you go through a process of looking around at a set of opportunities?

Rudkin: I’m sure I did, but I don’t remember them really.

Brock: Oh, I’m sorry.

Rudkin: Well, no I’m going to say over time, I generally interviewed at several places. But the ones I didn’t go to, I often don’t remember any longer what they were. I did interview at the company that was doing Smalltalk. Smalltalk had been spun off of Xerox PARC by this time. And I talked to them. And I talked to-- I don’t remember who else I talked to.

Brock: But it seems the theme was and perhaps what attracted you to-- well, I guess perhaps you could describe why you elected to go with Forethought.

Rudkin: Well, first of all, I knew Bob and Dennis already. I’d worked for Bob. I didn’t know Dennis very well, but through mutual friends we’d met. By the time I got there, it was a Macintosh implementation that was partly going. They had started thinking they were targeting Windows. And they did eventually. We did eventually target Windows. But by the time I got there, that was already postponed beyond because it wasn’t ready. And we were going to do a Mac version. I had seen a Mac. I’d used it a little bit, but really I didn’t know much about Mac except it was a very interesting user interface.

It was a startup. It was a company that had started up, got big, got small again. There were eleven people there when I joined having laid off a whole lot of others a couple years earlier. And what they were doing was publishing FileMaker and two other programs, which I don’t remember. But they were all three Macintosh programs that they did this model of publishing somebody else’s software, testing it, writing the manuals, doing the boxes and whatever else you had to do, making the disks, all those things. But it was a small company, which I liked. I liked the people that I met there when I interviewed, beyond just Bob and Dennis. It had a very interesting element to it, which is that the people within the company-- I said

there was only twelve people. Two or three of those were product support people, customer support people. And they had an 800 number that you could call. It was top notch support given that it was free, not even the phone call cost you anything. They had other employees of the company take turns at lunch times, noon times, so that the person-- maybe it was only one full-time person that did this so that that person could take lunch. So, I remember being on the phones and talking to predominantly FileMaker users. There was one guy who I’d heard about already. And he called while I was on, who lived in Dodge City, Kansas and used FileMaker to catalogue his rifle collection, his gun collection. I thought oh, that’s just great. That’s just a wonderful, small town use of a Macintosh and this software. And some of these people would call just to chat, which was the problem with having a free-- but I learned quite a bit because it was a small company. When FileMaker, which already existing, was going under a revision when I got there, one of my first days there I sat in on a project meeting where they were going over the schedule, where are we in the schedule, the testing, marketing materials, so forth and so on, mostly the code still wasn’t finished. And there was bugs being found and all those kinds of things as is usually the case. And I remember vividly Bob said to me, kind of an aside he said, “Now, if you look at this schedule, there’s always-- on a software schedule, there’s always a point that’s very important. It’s today. And the key thing to know is that the progress and such that happens after today is always going to be completely different from what happened before. It may look pretty bad when you look before, but the future is always rosy.” So, even though you might say our completion stuff is going like this over here, we still predict it will do this after today. Anyway, he was totally tongue and cheek, but that’s a fact about software people. We’re optimistic to a fault.

Brock: Well, one quick question is it seems almost that when you joined, the Forethought model is very much like the VisiCorp. You're publishing...

Rudkin: It was.

Brock: ...other people's stuff. It's in a genre and then, you're developing your, kind of, innovative thing.

Rudkin: Right. Now, I was not tied into that aspect of VisiCorp, but I was at Forethought. I don't think I did much, except sometimes attend these planning meetings, as I was saying, and being on the phone once or twice. But I was-- you know, a small company like that, you hear everything, and there were-- so I understood the process that they were going through of developing and shipping software, and that original FileMaker, which I guess it was called FileMaker 2, I'm not really sure, eventually, sometime that summer of '86, was released and it was a very good product, very much better than the-- I mean, it was a big success, as far as I can remember.

Brock: Well, in our discussions yesterday, I believe, unless you thought of something that you'd like to add, I thought we did a nice job of covering the process of creating PowerPoint 1, PowerPoint 2, and touched a little bit on PowerPoint 3. But maybe if we-- unless there's something else that, upon reflection, you'd like to add about that sequence of building-- making PowerPoint 1 and PowerPoint 2, I thought we could fast forward in your story to, you know, PowerPoint 2 is out. It's 1990 then, and maybe just describe-- if that seems okay to you, describe the rest of your trajectory with PowerPoint and with Microsoft, and I'm not even quite sure which year you decided to leave Microsoft.

Rudkin: Right. I left in February of '95. So after Windows PowerPoint 2 shipped, I mean, the first thing I did, of course, was to decompress. My wife and daughter and I went to Portland to visit friends and then we went to Hawaii to spend a week in Maui and such. But after that, we-- so PowerPoint 3 was the big topic, which had already begun, but was understaffed until PowerPoint 2 shipped. Because just about everybody-- PowerPoint 2 for Windows shipped. Because just about everybody was needed, in some capacity or another, development and otherwise on PowerPoint-- Windows PowerPoint 2.

Dennis Austin was in charge of the next PowerPoint. One of the things he had to worry about was, for the first time, we wanted to do a Mac and a Windows version of PowerPoint at roughly the same time, and I don't remember any of the details for that version. I mean, I know what happened by PowerPoint 4. But I believe that that was the point at which the Mac version might've been rewritten into Pascal instead-- sorry, from Pascal into C. But I was not part of that. What I did is I had a number of small-- quite a few small projects that I managed.

One of them was what we called Microsoft Draw. I mentioned yesterday that we had this Object Linking Embedding, OLE, that allowed you to have an application that fit inside-- not literally inside, but fit with a different application that you could have its creation, its data, its rendering, in the other application, and then, the first application would do the work, and the idea really was for it to look like it was really inside the second application. In general, I don't remember if we ever reached that very well. But before PowerPoint 2 for Windows had shipped in, let's say, February of 1990, development leads and managers were-- this is an aside-- often got together with our colleagues in Redmond in the other applications projects-- in the application division on the other projects, for a retreat or a get together to hash over various problems. This happened at least once a year, and this particular time, in early 1990, it was at Bill Gates' Hood Canal home. There were, let's say, 10 or 12 developers, I don't remember exactly, and Bill and some of the other bigwigs—Charles Simonyi was there and I think-- and Mike Maples, who was the head of the applications division, the vice-- I suppose it was a vice president. I don't really remember what they were called, but there was three people under Bill. Bill Gates was the Chairman and CEO. There was a President, John Shirley. There were really three divisions under that; Systems, Applications, and then the Finance and everything else. Steve Ballmer ran Systems. Mike Maples ran Applications. Frank Gaudette ran the others and he, unfortunately, died a few years later, very nice guy.

Anyway, the applications-- this was the Applications Division and Mike Maples was there, and we had gotten a lot of requests-- we, being the PowerPoint group, which was known as the Graphics Business Unit, GBU, had gotten requests to implement drawing for other applications, and Bob was not really too keen on the idea. But Mike Maples got Dennis and me aside and said, "You know, it really wouldn't be that hard, would it, to just put a small team together and sort of go for it?" And we thought about it, what the implications would be, so forth. We talked to Bob after we got back to Menlo Park, and we convinced Bob it would be a fine idea. So we had three developers, I think, very good developers, whose job was to take PowerPoint 2 for Windows, find the relevant code that does drawing that could be pulled out into a separate little applet, application, and then make that an OLE application. So that it could be embedded in Works, which was still working on their first Windows version, and Word, I believe-- I think Word already had a Windows version, but they were working on the next version, I'm sure. Maybe others, but, I mean, the nice thing is once you've implemented it correctly, then it would work with any of these other apps

who were implementing the host side of this OLE. So that's what we did, and there was three-- like I said, I think there were three developers. There was a QA person assigned, more or less, full time. There was a program manager. I can't remember if there was anybody else, initially. Four developers maybe. Anyway, and I was not developing on it, but these developers reported to me, and that was one of my projects.

Another one was the-- making sure the PowerPoint-- this didn't happen initially, but-- immediately, but making sure that PowerPoint 3 had a setup program that we wrote, instead of-- I don't remember how-- well, no. I do remember how we did the setup for PowerPoint 2, and it was pretty good, but it needed to be a lot more robust and do a lot more things in PowerPoint 3, and that was under my jurisdiction, and there was a program-- I guess, it was a Clip Art library, maybe, that I was the interface for, or somehow was involved. I don't remember exactly. I think there was a Clip Art program and then, Genographics supplied a lot of the Clip Art files. There was another-- I'm sure there was another one that I had, but I can't think now what it was-- oh, the guy-- the two guys who were working on Multimedia Window-- the PowerPoint for Multimedia Windows were also a project that I had. So I had these several different projects, and as I learned, you know, I spent a lot more time with the ones I cared about more. <laughs> This was not a good sign. But they all-- yeah, they all really, eventually, came to a successful conclusion, I think. There was one other I was going to mention, but I've forgotten now, again, what it was. Well, it may come back to me. So unlike Dennis, who had a team of people very highly focused on one product, meaning PowerPoint 3, I was doing a lot of different things. Oh, I know what the other ones. So this was, I think, when we had the person who would-- who did the translation between Mac PowerPoint format and Windows PowerPoint format. Yeah, that's who the other one was, and that was partly-- parts of it were pretty tricky, like getting pictures to translate and getting-- and doing-- figuring out what to do with character-- with character stats that were not just the 7-bit ASCII and so forth and so on. But the rest of it was pretty straightforward, in a way. There were things that were-- our file system-- our file format on disk that we had implemented, going back to Mac PowerPoint 1, was basically a bunch of, we call it B-files, block files, where-- containing blocks, file the contained blocks, each block of which was really almost an exact rendering of what had been in memory for data structure. And one of the things you may or may not know about Macintosh and Windows is that Macintosh based on Motorola is big-endian, and IBM, DOS, et cetera, et cetera, is little-endian, and so you have to convert numbers that are stored byte by byte in a different order in the two. And so-- but you had to know where were numbers and where were not numbers. You couldn't just take every pair of bytes and swap them. So that's the kind of product-- but, you know, it was not rocket science, as they say.

Brock: And could you talk about your decision to leave? What...

Rudkin: Well, so PowerPoint 3 went through 1992, I guess. I don't remember when the Mac version shipped but I know the Windows version shipped in the summer or so. Bob Gaskins left a little later in '92, I think maybe before the Mac PowerPoint 3 shipped. I could be wrong about that. Things were kind of, you know, like, "What are we going to do now?" I don't mean products, but I mean what am I going to do now? I was the-- when PowerPoint-- PowerPoint 4 was a rewrite into C++ sharing, where possible, Mac and Windows code. I had the job, amongst others, of managing-- or leading the development of the Mac version, after the Windows version had shipped. This is-- there's this-- the theory was it would work

because you kept the Mac version and the Windows version working all along. It's the same code base. The practice, however, is that when things get tight, you stop learning about the Mac code. You don't even compile it anymore. This is the way it was going, and you don't-- so when you get the whole Windows version finished-- and the way you make things work both ways is a combination of code that is computer independent, machine independent, and code that is only visible in the Mac code or only in the Windows code, and it's either through compiler flags or different files or some other technique. You include the Mac code or the Windows code, depending. And in theory, you have some fairly low level things that are different on the Mac and the Windows, but everything that calls those is independent of the machine. That's the theory. But it's not what actually happens, and since a lot of this-- so you have a lot of these what in C are called pragmas where you'd say, "If-- you know, #if Mac this, else this, you know--" or it may be the other way around. But if nobody has kept that code working and even tried it for many, many months, it probably doesn't work, and so a lot of what we had to do was go back now, and it probably took us from summer of '94 through late '94-- I don't know. Not that much. Four or five months, I guess, to get that finished. It was the first Mac development I had done since PowerPoint 2, and that was good. But it was always, kind of, the tail. You know, the real product was the Windows version. We had to have a Mac version. It had to be good. It had to share files correctly. But we're not going to sell as many of them, so we don't put as much effort into it or as much attention to it, or however you want to say it. And at the same time, there was also a guy in my group who was doing the Kanji version of Windows PowerPoint. He had been on the standards committee, or whatever, for Kanji character sets, and I don't remember anymore whether that was Unicode, but I think it was. Yes. He knew a lot about Unicode; that's right, and so he was making Windows PowerPoint 4 Kanji work, which was, you know, a reasonably hard task. But, you know, it's a lot-- it's not hard to get the French and Spanish and German versions, once you have the English version. If you've done your job properly, it's dead simple. You probably have overlooked a few things, like you didn't realize what happened when the text doubled in size in this dialog. You have to fix those things. But by and large you've already taken-- from day one in Macintosh and Windows, you've already taken all of the text that the user will see out of the code and put it into separate places, where it can be modified by translators. That was the-- we did that, by the way, with VisiOn going back to that, except it was a little harder because we hadn't been meticulous about-- we didn't have as good a model as the Mac did of separating text out of a code, and it's not just text. It's also things like metric versus non-metric, but we'd already done that from day one. You could switch between metric and English in-- I'm pretty sure in Mac PowerPoint 1. We supported A4 paper and few others like that, whatever the printer supported, basically. But anyway, the changes within the broad set of Western European languages is pretty simple, and even Cyrillic isn't that hard because it's still a fairly small character set. But when you go to Kanji or the various Chinese character sets or--those are the two I am familiar with in this context--you've got a lot of other things to worry about. Arabic is right to left, for example. I don't have any direct experience with that but I know it's tricky. So that was another project that I was supervising at that point. But, you know, eventually I decided I really wanted-- I'd been working on PowerPoint since 1986. It's now summer of '94-- from '86. It's now the summer of '94. I'm working on these other things. But I started to say to myself, "I've worked on PowerPoint long enough. But I don't want to leave Microsoft. What else can I do?" Well, there was no other Microsoft development in the Bay Area, zero. I interviewed for a post in a database group in Redmond that was looking for somebody to do a, I'll say broadly speaking, like FileMaker. It's a Graphical User Interface front end for data entry and printing and all those things, where you could design your forms. But we had-- by this time, '94, we'd lived

in the Bay Area since '76. Our daughter was a freshman at Cal. My wife just really did not want to move, and I could understand that. I only kind of wanted to move. Seattle I like, except for the rain. But I think could've put up with that. But well, we have too many friends down here. So I left in February of '95, after some of these projects wound down.

Brock: And maybe we could just briefly survey some of, for you, the most meaningful parts of your activities since that time.

Rudkin: Well, let's see. There's two or three things. First, there's family. My daughter graduated from Cal in '98, stayed in Berkeley, eventually met a guy and they're married now. They have two sons, an eight year-- a nine year old and an 11 month old now. Those are a big part of what we do. I am a soccer referee, and have been since just before I joined Forethought, in fact-- no, sorry. Just before Microsoft acquired us. My daughter played soccer, AYSO, American Youth Soccer Organization, beginning when she was about eight, and a couple of years later or three years later, I became a referee and I love it so much that I played ever since, even though she's no longer playing. I mean, I've refereed ever since. I did other kinds of youth soccer and lots of adult soccer, men and women. I did high school-- I still do high school and I did college soccer for about six years at the junior college in Division Three levels. But I still do that to this day at high school-- high school season here is a winter sport, and this past winter, I had probably, on average, three games a week, boys and girls, varsity and junior varsity. So I have to stay in shape. I do a lot of running. Other than right now, my Achilles is kind of bothering me, so I've been biking instead. In addition to refereeing, I also instruct referees within AYSO, quite a bit. I haven't done any of that recently, but for some years, I was instructing even at the top level of youth referee, you know, referees for youth games.

What else? We travel a lot, overseas and here. We bought a home-- well, we spend a lot of time at the University of Kansas doing-- because we're on some advisory boards. We're both on the advisory board for the Natural History Museum and Biodiversity Institute. I'm on the board for the International programs, due to my study abroad back then. My wife is a trustee on the endowment association. We've got some other things we do. We have a lot of friends there, so about five years ago, we bought a house there, about an 1875 Victorian, and part of the time, we have a-- well, my niece was living there at one time, when she was undergraduate, and now she isn't so right now we don't have a tenant. But we use it for ourselves often enough that it's a pretty good deal. It's very nice to not have-- not just not being at a hotel. That's one thing. But being able to cook your meals, being able to not have to transport a whole lot of clothing and all those other things. It's nice to have a place that you can go to, and you already know it. It's not a motel or a hotel room. We've traveled overseas. We did a biking trip in 1999 in Southwest France, through a local bike shop here. That was wonderful. I hadn't been back to that part of France since my study abroad year. We went to China in 2005. Kind of the usual things, but the thing I really wanted to do then was get on the Three Gorges and Yangzi River before that-- I wanted to go before the dam was closed, when it was completely at its original water state. Unfortunately, we couldn't due to some health problems in my wife's family. So instead, we didn't go until 2005. The bridge had been-- or the dam had been completed in '03 or something. But the water hadn't risen very much.

Brock: Were you ever lured back into technology or software?

Rudkin: Well, I did-- yes. I did one project, very short-term project, for a friend of mine, who was working at a startup and wanted some help with the UI design for his project. I got involved in a children's web-- well, it was called "World at Play," initially, and then "E Play." But it was kind of an online game playing, trying to be a safe environment. I got involved both as a development technical advisor, if you will, as well as, as an investor. It turned out to be another one of those projects that was trying to do way more than they could do with very little money, and I spent a lot of my time with the founder, visiting venture capitalists and by and large, we never succeeded at any of those VC visits. But went to Boston and New York to talk to some finance people, and we showed the product in London, one time, at a show. I think it was-- I think California was, the state, I think, had supported having these businesses show their wares at this big Queen Elizabeth conference hall near Westminster. Those are the only-- I read a lot. But I don't actually program or anything like that.

Brock: Well, I thought maybe we could spend-- and a little bit mindful of the time, but I did want to see if you might reflect on just programming and software. Of programming, as an activity, and as you see it-- or making software more broadly as an activity, and some of the characteristics that you think are helpful for people to have, to engage in that, and also just throughout the course of your life. From when you first became involved with programs, as an undergraduate-- and software, to the present. Just a reflection on the change in software, its place in society, just any thoughts you might have on those two domains.

Rudkin: Okay. So-- all right. Let me think about this. Software itself, of course, used to be a niche field. There was a lot of it going on in businesses, as I mentioned before. But not really very many people got their-- made their living creating software or on computers, for that matter, and that certainly has changed. It has particularly changed since the web became prominent, and I've never written any web software, so I can't really speak to that firsthand. But even before that when the Macintosh and the IBM PC line of computers, and before that even the Apple II came out, there were a lot-- you know, suddenly, you saw computer stores where you would go in and there would be games or other pieces of software on the shelf. I can remember--there's been several that no longer exist--that I would go into and I'd go, "Wow. This profession actually now is seen by the masses. It's no longer just a behind-the-curtain kind of thing." There was a bookstore--it may or may not still be there--over by Lawrence Expressway in Sunnyvale, called Computer Literacy, I think is what it was called. When I would go in there to buy a book on the IBM PC or this, that or the other, and it just always amazed me that there were so many books-- even then. This was '86, let's say-- no. Even before that, '84. There were books in here that I didn't think very many people would even care about, you know? So that's a change. You go into Fry's which was started in, I don't know, '85 or so. I first went there in '86 with Dennis, when we need to get a cable for the printer that we were hooking up to this network that we had in the Forethought building, and the cable had to be constructed because there wasn't-- they weren't off the shelf, and there were people at Fry's who could do that for you, if you told-- if you had the specs, and "Wow. Look at all this computer stuff," and this was a hobbyist store, really. I mean, this was before it became a big, you know, appliance and everything else, and full computers. I mean, it was, but it-- that was not its bread and butter in those days. That was pretty cool.

So certainly, the number of people building software and also the number of separate products, if you will-- it's hard to say what is a product and what is a piece of another product. But the number of those is

really high now, and I think that's mostly a good thing. Although, much of it isn't any good, you know, but that's okay. The market will figure out which ones live and which ones don't. I'm pretty old school about a lot of things in software creation. For example, I'm a very big believer in planning ahead and doing a structure to your design and to your implementation that can be understood, not just by yourself. I mean, if you're doing a project that you know will only ever be you, that's one thing. There's a portion of Frederick Brooks’ book, "The Mythical Man-Month," which he wrote on the history of creating the system 360, I believe, for IBM, and talking about many of the things that either went wrong or could've gone wrong that he learned during this project, and one of them that I remember vividly is kind of a three-by-three matrix. In the upper left is, "You wrote this code for yourself," and it does whatever it does and that's fine, and in the second, maybe, is "Now you're sharing it with some other friends who also want to use it," and they find all sorts of problems that you didn't encounter because you didn't do those things or, you know, they wanted to do some things that you didn't care about or they had rough edges that you didn't care about, but they do care about, and then there's something that you're actually going to make as a product that you're going to sell, and these are probably-- order of magnitude increases in the work required to do them, and the other dimension is-- and I actually forget which was across and which was down, but bear with me here. The other dimension is, "What does the software do and how many things does it have to worry about?" The top is basically a program that does a fairly small number of tasks. The bottom is a system that has to be able to run other programs and have an interaction with other programs of unknown origin, et cetera, and in the middle is something, which I can't tell you right offhand what it is. But the point is, when you go from up here to down here, you're probably, you know, six orders of magnitude more difficulty. <laughs>

Building an operating system for general use-- you know, the guys who built Unix, for example, for themselves, that was pretty hard. When they had to support a whole lot of other people, now that's a different thing entirely. So my point being that if you are building a piece of software that is convoluted code or uncommented or the structure either isn't there or is not apparent, but it's there, how the heck is anybody else going to be able to take it over later? It just isn't going to happen without as much work. Many times, people will just say, "Well, I don't know how that works. I'm going to start over," if it's in that shape. Structured programming was the buzzword in 1973 and before, when I started doing this in classes. You know, I even took one course where we did some of the work in Fortran, and the instructor showed us how to do structured Fortran. I thought, "Really? Structured Fortran?" But it-- you know, there's something to it. The algorithms are very structured, even if the language isn't. I mean, if you're going to build loops, you know, you indent to such a way that it tells you. It can tell you where the loop started and the loop end-- I mean, there is no loops in Fortran, except go-to's, back then, at least. I haven't seen Fortran in 30 years.

So I'm a big believer in doing it right, putting in as much effort as you can to build a foundation that you're going to use that will support the things you have to do, but also allow other people to support it, to adapt it, to port it to another machine, whatever the case may be. If you're doing anything with a commercial-- you know, with an eye toward it being used by people, other than yourself, and there's a lot of ways to do that. You know, there have been various things that have come and gone in vogue. But they all sort of have this same, you know, separating function from implementation. "What does this thing do? If I call this function, what's it supposed to do for me?" That's all I should have to know. I shouldn't have to know how

you implemented it. That kind of thing, you know, data hiding or implementation hiding or whatever you want to call it. There's a lot of different words that have been used, over the years. Now, I'm not trying to say that that's not the way people do things today. But I don't-- I doubt it's any more the case than before, and it might less the case.

Brock: Well, maybe we could close, just by my asking is there anything that was in your mind about wanting to discuss that we haven't gotten to, or something that's occurred along the way that you can think of?

Rudkin: Maybe there have been, but I can't think of them now. <laughs>

Brock: That's always the case. <laughs>

Rudkin: I know. Well, I would just like to close by saying, first of all, I very much enjoyed being a software developer for 20 years, and I very much enjoyed not being one for the next 20 years. Because I got kind of burned out. I got kind of tired of deadlines and so forth and so on. But more than that, I'm able to do so many more things now, due to Microsoft. I mean, if I hadn't-- if Microsoft hadn't acquired Forethought, I would not be where I am today, by any stretch of the imagination. So I really owe everything I can do today and the fact that I could retire when I was 44 to having worked at Microsoft. Now, the other thing I'd like to say is that I'm very honored to have been selected for this interview process, and I think this has gone very, very well and I thank you for that. And I thank the museum for that because they supported it, too, very well.

Brock: Absolutely. Well, thank you very much, Tom.

END OF INTERVIEW

