Oral History of Kenneth (Ken) Kolence

Interviewed by:
Luanne Johnson

Recorded: September 16, 1996
Berkeley, California

CHM Reference number: X5400.2009

Table of Contents

Kenneth (Ken) Kolence

Conducted by Luanne Johnson

Abstract: Kenneth Kolence, one of the founders of Boole & Babbage, the first software products company in Silicon Valley, talks about his life-long interest in software engineering and management of the software design process. He describes his introduction to computers at the University of Illinois, his work as a computer programmer and operations manager while in the U.S. Navy and his subsequent career as a manager of the software development process at North American Aviation and Control Data Corporation. He talks about his early and on-going interest in measuring computer performance, which led to Boole & Babbage's successful products, Problem Program Evaluator and Configuration Utilization Evaluator. He describes the challenges in marketing software in the early days of the industry and the impact of IBM on competitive markets for independent software vendors.

[Editor's Note: This interview was conducted at the restaurant Skates On the Bay in Berkeley, CA.]

Management of the Design, Development and Implementation Process

Kenneth Kolence*:* I think one of the most important contributions of the software field will be in generalizing the concept of how you design things, design descriptions. And I think we'll learn how to design things much more rapidly than we do even today. Today we do a fantastic job with the CAD stuff for all the hardware.

Luanne Johnson: Compared to several years ago?

Kolence: No one else was faced with the issues of a design process that we in software were faced with from the beginning.

Johnson: Right. That's one of the themes that's coming out as I do these interviews. When people were writing very ad hoc programs with each computer installation, they would just do it until it worked. They didn't worry about the design function because they never were going to take that and use it someplace else.

It wasn't until people began to think of software as a product that they began to start front-ending the development of software with some real thought into how it was going to be designed because they had to make it flexible enough and robust enough to be able to work in different environments.

Kolence: That's not completely true because on the commercial side of things the idea was originally that you had to have systems analysts who were problem domain specialists. And then there had to be a way of communicating what they felt was needed to the programmers, the people who built the software.

When it was a one-to-one mapping that wasn't too hard to do. But at North American Aviation… let's see, I took over the commercial programming group in about 1962 and I began to work on the problem of structuring the design process. I had always been interested in the design process. So, it didn't happen just because software began to be sold as a product.

I got that working very well and straightened out a lot of things. I was the first person to separate maintenance from development activities. And the reason was we were always late with the development projects because the best people were also responsible for maintenance on other projects.

So I was the first person to separate the two. That was done at North American Aviation, in the Aircraft Division, probably in the first part of 1963. We really straightened things out.

Then also at the same time, I did not just turn it over to operations but hired a guy whose job was to work with operations, and to understand precisely what they needed to run production. We actually had a group who would take the software from the developers and then build exactly the documentation that the operations people needed. And we trained the operations people in using it. So we put together a complete life cycle view of the whole thing.

I left North American in early 1964 for two reasons. First of all I didn't like Los Angeles. I was walking around with a roadmap on my eyeballs all the time because of the smog. Even within the building you couldn't see very far because there was so much smog in the building.

But the more important reason was that when I was in the Navy, I had been responsible for every area within a normal installation. Operations, systems programming – I I was one of the first systems programmers – and then development. So I wanted to go to a company where I could see the other side.

Introduction to Computers at the University of Illinois

Johnson: Is that what got you into the whole field?

Kolence: No. What got me into the field was the beginning semester of my senior year at the University of Illinois when I had an extra space on my course schedule and my advisor said, “Why don’t you take this new course in computers?” And I said, “Okay.”

I didn’t know what the hell computers were. But it sounded like something I might be able to handle. I think it was running the Marchand computers, you know? After about two weeks in that course I finally raised my hand and I said, “You mean computers don’t do algebra? They just do arithmetic?” Because we were busy trying to design add-ins.

I didn’t care much for the design but the programming side I fell in love with. So I’ve actually been programming since the beginning of 1953.

Johnson: Wow. There weren’t too many people programming in 1953.

Kolence: On the IILIAC I. They had just brought the IILIAC up for use by students. The IILIAC is an ORDVAC twin. They delivered the ORDVAC and while they were building it they built the parts for the IILIAC. ORDVAC was Ordinance Research Computer or something like that. They were built at the University of Illinois.

Johnson: One statistic I ran across when I started doing research on software history is that in 1955 the total programming stock, the total number of programmers in the U.S., was 1,200.

Kolence: Oh, really?

Johnson: Seven hundred of them worked for Rand on the big SAGE project with the Air Force. Does that sound right to you? That would have meant there were maybe another 500 programmers working on everything else.

Kolence: I always said I was one of the first 3,000 programmers. But that was a lot. I really thought it was maybe the first 1,000.

Johnson: I think you must have been.

Kolence: But don’t forget that there were programmers for the UNIVAC then.

Johnson: Right.

Kolence: There were the IBM 701, 702, 704 and 705. The 650 didn't come out until later. The 701 and 702 were Williams Tube memory machines. The first memories were cathode ray tubes. And they were called Williams Tubes.

Johnson: Oh, they were? Okay.

Kolence: You could shine a beam on the surface and they had little capacitors on the outside that could detect it and then they would regenerate continuously. The 701 and 702 were Williams Tube memories. They were terribly unreliable. But fast.

And then they came out with the 704 and the 705. I think they came out with the 705 first. The 704 was a vacuum tube machine but it had better memory. I think it must have originally had Williams Tube memory and then they went to core.

But the IILIAC I had Williams Tube memory. And that was brought up in the fall of 1952. Wait, did I say I started in 1953? It really was 1952 was when I started.

Johnson: Okay.

Kolence: Because I graduated as an undergraduate in 1953.

Johnson: Okay. So go on from there. What was the sequence of events that finally led to starting Boole & Babbage?

Kolence: Well, I was in the Navy first. Because when I got out of school I had my Masters in math with a…

Johnson: You got your Masters from Illinois, too?

Kolence: Yes. With a major in analysis and a specialty – they didn't give anything else – in logic design and computers. Programming and logic design. I took more Boolean Algebra than anybody deserves to take. But that's how they designed all the computers in those days, basically by writing Boolean equations for the circuitry. Which was a key fact in all the things I've done.

Johnson: Okay.

Programming and Operations Experience in the U.S. Navy

Kolence: So I joined the Navy and I got married. But I was always going join the Navy anyway because I grew up next to Great Lakes, Illinois, and I just assumed I was going to join the Navy. I got stationed at David Taylor Model Basin, the naval ship design center right on the Potomac just past Cabin John.

Johnson: Yes, right.

Kolence: I got stationed there because they had a UNIVAC, the only computer the Navy owned at the time, except perhaps for Naval Intelligence. I was supposed to go into Naval Intelligence but they had closed the roll. I'm just as glad I didn't get in.

So I joined the Navy and I started as a programmer. But the UNIVAC was falling apart. Literally. So this other Navy guy who was an enlisted man but a EE graduate and I fixed it up and installed all the preventive measurement procedures and stuff like that. And we got it humming so that by the end of the year, it was good as new as far as reliability was concerned. I even had to replace one of those old mercury panels in the middle of the night with a gas mask and in my skivvies.

But that was not what I wanted to do so I got myself transferred up to Philadelphia Naval Shipyard where I worked with Grace Hopper.

Johnson: Oh, did you?

Kolence: And a couple of other people. A lady named Betty Holburton. She held the copyright on the UNIVAC op tools.

Johnson: No kidding?

Kolence: Yes. She designed them. She worked there. Commander Hopper came down to David Taylor to do her two weeks active duty so I got to know her there. And then I moved up to Philadelphia Naval Shipyard and worked on installing a UNIVAC II.

Maintenance on the David Taylor Model Basin computer included operations because operations and maintenance were never separated on UNIVAC I. You needed to understand the guts of the machine to operate it properly.

At that point in time, I designed the production operation side of things for the shipyard. They were going to do payroll and stuff like that. And they were writing it in B-0 which is the predecessor of COBOL. B-0 was the first language equivalent to Fortran for business processing.

Johnson: Okay.

Kolence: And later it evolved into COBOL.

At any rate, I worked with Commander Hopper and trained the operators and did all the documentation and whatever was needed to keep the machine as utilized as possible. And also to avoid things like bad tapes, etc. So I set that all up.

Our delivery of the UNIVAC II was delayed for a long time due to conflict between the ERA division of Sperry Rand and the Philadelphia people, the old EMCC [Eckert-Mauchly Computer Corporation].

Johnson: They had acquired ERA, right?

Kolence: Well, Remington Rand, the round punch card people, had acquired EMCC. So they had the UNIVAC I and then they also acquired Engineering Research Associates. The guys from ERA designed the first Control Data machine on Remington Rand's time.

They bought the EMCC people and so the UNIVAC II finally came out and was nothing more than UNIVAC I with core memory. That's what gave IBM the opportunity to get into the commercial business because Rand was two years late in delivering a new machine and they didn't have things like channels. It was pretty slow.

I estimate – and this is just a guess thinking back on cycle times – that the UNIVAC was just a 200 instructions per second machine. Roughly. When you took into account all the time it took to do I/O, it was slower than that.

So at that point I had two kids and was desperately poor and I wanted to go back and get my Ph.D. I had an assistantship from Jay Forrester at MIT but I just couldn't afford it.

Move to RCA and BMEW

I mean, I'd been working at two jobs all my life. Three jobs a lot of the times. When I was 17, my family broke up and I was completely on my own. Actually I was more like 16. So I went to work for RCA on the ballistic missile early warning system. BMEW. The big radar.

Were you around the New Jersey area let's say in the 1960s, 1970s?

Johnson: No, not much. I traveled there occasionally but didn't spend much time there.

Kolence: Did you drive past one great big radar dome near Castlefield? On the New Jersey Pike?

Johnson: I may have.

Kolence: Well, that's the one I put in. I put in the 709 and I was in charge of operations. Keypunch and everything except programming which meant that I got the first operating system from IBM called IBSYS; it was the first operating system that they delivered.

I'll give you more history on that. Owen Mock was the chief operating system architect for SHARE but he worked for North American Aviation not for IBM.

Johnson: Right. Because the users felt they needed an operating system and SHARE members got together and designed it

Kolence: SOS it was called. SHARE Operating System. The SHARE members were Douglas Aircraft, North American Aviation and some other groups, in the LA area primarily.

It was at RCA that I decided I wanted to learn how to measure the performance of computers.

Johnson: It goes back that far?

Kolence: It goes back to 1958-59. Well, it goes back even further. On the IILIAC I they had a slave Williams Tube which gave you one bit. So you could look at that tube and see if it was going all over or going into a tight loop or what have you. And so it was a way to understand if your program was bad or not.

But then I read this article in – I think it was *Factory* magazine. I always was interested in industrial engineering and they talk about sampling to find out what kind of work was being performed where on the factory floor. Just like population sampling when you do polls.

And I said, " I can do that for a computer." But the problem was there was no timer involved to speak of. But that's where I started working on the problem. I got very interested in that because we had all standalone machines. The only thing you could do is find out where the problems were for each program.

<u>Move to North American Aviation</u>

So then I went to North American. I got tired of the snow in New Jersey. I had no relatives there. And I didn't want to go back to Chicago where my relatives were because my relatives *were* there. But that wasn't the only reason.

At RCA I got active in SHARE. And Jack Strong from North American Aviation, who was one of the main founders of both GUIDE and SHARE, and I hit it off well. Jack wanted me to take on the "SHAREmanship" committee which was the official greeter for new people.

I'm a born host. Always have been. But I read through what the "SHAREmanship" Committee did which was welcome new people and do whatever else is necessary to satisfy the needs of the membership. Ah-hah. A loophole. So that was good because SHARE was focused on all real dirty systems programming work and I opened it up to management issues, to modeling. Anything anybody wanted to form a subcommittee for, go ahead.

I was the program chairman after that for a while but that didn't happen until I went to North American which was January 1961 after the BMEW system was finished. I had gone to Los Angeles to a SHARE meeting and I liked it. Strangely enough it was one of the few clear days, clear weeks etc. But I really thought it was nice and I wanted to go with one of the big companies.

So I went to North American and I was sort of a staff guy for a year then the guy in charge of commercial programming quit and they put me in charge because I had lots of management experience. So here I was just turned 29. I should go back and put another thread in place.

When I was at the shipyard in 1957 or early 1958, the thought came to me that we really ought to be able to write equations to describe any business process. And the reason I thought that was that it occurred to me that, since all computer instructions are originally defined in Boolean equations, therefore each instruction could be described by an equation. Therefore, as an existence proof, not as a constructive proof, one should be able to write equations and you could theoretically write equations for any program you wanted to, any program that represented what was being done in the real world.

So, in that sense one should be able to do it in algebra. While I was at RCA BMEW, I spent a lot of time trying to follow through on that. The equations I wrote were like ten feet long. So I knew that wasn't going to work. But I never got rid of that thought and I still haven't gotten rid of that thought.

As a matter of fact, my latest work is still very much directed towards some of these issues. I sent one of my papers to one of the foremost practitioners of this kind of algebra describing programs and operations, Heinz Teeloff at IBM. And he said he was astonished that I was able to see it that well for the time.

So at that time I was still thinking about the design issues. How do you design things? We really should have an algebra or something like that. When I worked at North American, I came across decision tables. Decision tables gave me a lot of things I could do to write the equation. They didn't tell me what kind of an algebra I had to have. They didn't give me the standards operation, things of that nature, but they gave me a lot of what I needed.

At that point, we had a 1401. A 1401 is a real little machine.

Johnson: I learned how to program on a 1401, so I know.

Kolence: We had lots of them. They ordered 42 or something like that. Basically they replaced most of the tab machines.

But when we were writing programs on it, the programmer would sit there and I started testing out instrumentation sampling. I would have him push the stop button, record where it was, turn it back on, push the stop button. A hundred, a 120 times for a program. And then draw the histograms and then look at the code and see if, in fact, they correlated. I was very careful to verify everything.

And then later on we got a 7090 at Autonetics which was another division of North American. And a guy named Jerry Geritsky had a timer on the 7090 which cost $1,000 a month. Back in 1963.

Johnson: $1,000 a month was a lot of money then.

Kolence: I wasn't earning that much and I was in charge of everybody.

Johnson: You could buy a couple of programmers for that.

Kolence: So we tested it out on some big Fortran program. It worked like a charm but the programmers didn't like it. I'll come to that later.

Anyway, I'm busy laying out the design methodology, writing procedures, making sure that everything we did, anything that anybody produced was in directly usable form. Because the rule I had was that to produce anything to give to somebody else to do something with it should be directly in the form that they can use it. So many documents had to be redone. We integrated that whole process and things really hummed.

But as usual I got bored. There are a number of threads that run through my life one of which is I'm very good at getting something working and staying with it for a year or two after it's working and then I get so bored I just quit.

Johnson: I call that low boredom threshold.

Kolence: I don't know what it is. It's just I'm good at building, good at creating, but I'm not any good at just hanging around and keeping house.

Johnson: I'm that way myself. The way I think of it is that I need to always be on a steep learning curve. I need to be in an environment where I'm constantly having to learn what I'm doing. The minute I get to the point where I really know what I'm doing, I get bored with it.

Kolence: That's about me too. I've always been that way and it's one of the worst problems you can have, especially as you get older.

Johnson: Everybody else is settling down and I'm saying, "Oh, it's time to start something new."

Kolence: Same here. Exactly that.

So I went to Control Data and I was in a department that was called the Analysis Department for no good reason except it was a dumb name. And a guy named Clair Miller was the managing director of all the software for Control Data Corporation.

But first let me finish North American. We got all the organizations running smoothly. I had people that were quite capable of doing the job. We had procedures manuals and all that sort of stuff so, in fact, we had a pretty doggone good design process.

I left partly because I was bored with that stuff and wanted to go on to the other side of manufacturing. I'd been Vice-President of SHARE at this point in time. Now, the general rule was if you get to be president or vice-president of SHARE, when your term was over, you get a job with IBM. I couldn't stand the thought of working for IBM because they really did sing songs.

Johnson: I know. I know people that did that.

Organizing the Design Process at Control Data

Kolence: That just wasn't my cup of tea. I couldn't exist much less thrive in that environment. So I didn't go to Poughkeepsie. But Control Data Corporation was giving IBM a run for its money early in the mid-1960s with their 6000 series computers. The Seymour Cray design.

And so, because of my contacts within SHARE, I pushed myself on them and figured out that their headquarters for all software development was in the Stanford Industrial Park in Palo Alto. So I went to Control Data and started designing a database and a report generation program based on the principles of mathematical equations and that stuff. And they had another guy doing the same thing.

We both finished our descriptions about the same time and he got the job to proceed with the implementation. I was so pissed off. What I was pissed off at is they didn't have any way to evaluate along the way, to initiate things. I didn't know what they were going to evaluate it on.

So I went up to Clair and I told him that if he gave me a chance, I'd straighten up the place. Control Data had a terrible reputation for not delivering software on time, with too many bugs and documentation errors, and there wasn't any way to know when stuff was coming out. There wasn't a formal way to announce the software. No way to know all the different possible equipment configurations that the software would work on. All that junk.

Johnson: Basic organizational stuff.

Kolence: Yes. Real basic. So I was so peeved about this, that I really needed to get the damn organization straightened out so I could do what I wanted to do. So he made me head of this Analysis Department. With a couple of different jobs.

But the main job was to run the design process in a staff capacity. He was the guy that had the final okay on everything but I was the guy that everybody else had to pass through. And he gave me a group of people to work with the various directors. There were like four or five, one back in Minneapolis. They were pretty good people. Most of them became vice-presidents of Control Data.

So I put everything on a PERT chart. Laid out what we'd call a meta-model for the project now. This was in early 1965 or late 1964. I'm not sure which. I wrote up the procedures manual, explained what had to be produced, who had to review it, how it had to be initiated and budgeted. The review procedures.

So we had the meta-model and each project had to be reduced to whatever details needed to be made but they all had to report against the same milestones.

We set up a small group, one or two people, to manage the distribution of documents but also to run the PERTs every week. Each week each project in each division would run their PERT reports for the division manager.

They would use that to determine where the problems were and how to fix them. And once a month we would have a meeting that would review the problems that had actually occurred and the projected problems and how to fix them.

And once a month in a different meeting two weeks later, we would review requests to initiate projects. Everything went through me. All these people that worked for me were really working for department managers. I said, “You can do whatever they want you to do but make sure that everything gets reported and reported in a standard way. This is the standard way.”

So that’s what we did and we put in the standard things like product availability dates initially shown as the first half and then one quarter out and then finally with an actual date. I was the first person to my knowledge to do that on the West Coast.

There were probably things like that happening at IBM. As a matter of fact, some of there terminologies I invented were the same as IBM had like External Reference Specifications. Things like that.

The External Reference Specifications were what were tested against. We gave that to a testing group. They are just like getting a manual from Microsoft or somebody today. Each function and each operand and what happens. The internal design was up to them but the company owned the external characteristics.

So we got that tied in with everything that everybody needed. With the PERTs, we could identify where the problems were. When we first started up, there were hundreds of things that missed, late, things that hadn’t done. We cleaned everything up and within a year we would have only two, three potential problems per month because we were only looking at it once a month. The other people were looking at it every week. They were taking action and my guys were loyal to their director but they were honest with me. It really was running smoothly.

And then the other thing I did at that point in time was I started insisting that everything have product numbers. This was end of 1965. Each Fortran product had a different product number. You couldn’t keep them apart. There were several different Fortrans and things would get shipped that were wrong.

So we put in a product identification numbering scheme and everything had to be right there to be authorized. Everything was tracked on that including the PERT information. Everything had to have that number on it.

I got a lot of flack about calling software a product but nothing compared to the flack I got later. The Control Data machines had timer interrupts on them. So as soon as I got appointed manager of the department, there were a couple of guys that weren't doing too much. I laid out the design for a sampling monitor and they put it together for me.

The first time they ran it, they looked at it and said, "It's completely wrong. There's a bug in it because there's only one place where this program ever executes from. It's just flat and then one bar and flat." This was the Assembler. But it turned out to be correct. It was waiting for a tape to rewind. Write. Rewind. Write. Rewind. So that was a real break-through..

We ran a full set of experiments determining what we had to do to meet the criteria for random sampling in a computer program. We had a random sample generator and we put in a range of times we wanted. Say we wanted to have one a minute or one every ten seconds. Then we'd say one every ten seconds plus or minus two seconds and we'd randomly generate a number there. And gradually pull that together until it was once every ten seconds. You couldn't tell the difference. The numbers were the same. So we did all the proofs and I said that it was available if anybody wanted to use it.

I got some real nasty letters from the programmers. "Why are you doing this? We're professionals. We don't need anybody to measure our programs." The same attitudes in spades that there are today.

Well, it turned out that Control Data had contracted to deliver a bunch of 3300s, a small machine with a 1403 printer simulator. It was supposed to be twice as fast as the 1403 but, at the least, just as. It didn't work that way. They weren't getting any money from their customers because all these things had been shipped but there was a guarantee that it would perform as promised.

So they were in deep trouble. Actually I think some of the 6000 computers also were tied in with this guarantee because no money was coming in. And here I'm sitting with my sampler ready to go.

So they put a team of people on that for about 45 days. Sixty at the most. Had it tuned beautifully. Running at least as fast on all the workloads that they could get as the 1401/1403 combination. And faster in some cases.

Well, I never heard another bitch from the programmers. But they wouldn't use it. But the neat thing was that back in Minneapolis, they noticed a dramatic improvement in delivery and reliability and how neatly everything was packaged. The main problems then stopped being software and started to become hardware delivery issues. So they started using my techniques for hardware.

Clearly you can't transport one of those things into a new design method without making some changes and so they wanted me to come out there and run that show. But I wasn't about to move from Palo Alto. And about that time they also decided to consolidate all the programming back in Minneapolis. That was early 1967.

Johnson: Okay.

Kolence: This design methodology by the way was starting to be known in the area. If you look around here most of West Coast companies began to use what later became waterfall design methodology. Most of the people, like at HP and places like that, came from Control Data and introduced that methodology so it spread from Control Data.

Johnson: Okay.

Kolence: Not to say that IBM's wasn't similar. It's just mine was a lot simpler because we were dealing with smaller staffs, not ten thousand programmers. So that level was more appropriate and it worked pretty well.

I got kick out of the fact that the last issue of *HP Journal* is all about software quality and a guy named Grady who is one of the top guys in quality showed a fishbone diagram. Do you know what a fishbone diagram is?

Johnson: Yes.

Kolence: And up there it said External Reference Specifications. So I had to laugh that that term was still in use.

In the 1960s and 1970s mostly there wasn't any competition in the sense of too many different products doing the same thing. There was just the policy of the IBM salesmen not to give any money to anybody except themselves and to maintain their account control.

Johnson: I started my business in 1971 selling accounting products, payroll, accounts payable, etc. And I remember very clearly a sales call that I made in about 1975 when I realized that I was selling against other software vendors for the first time as opposed to selling against the idea that the customer's staff could do it better themselves. And my reaction was, "Great! This is wonderful!"

Because then I could compete on function and price rather than competing against the idea that they could do it themselves. So from 1975 on it was a matter of positioning myself against the other software vendors. But it was as late as 1975 before that became the usual sales situation.

Kolence: When David [Katch] and I quit Control Data, everybody said we were crazy to start up our own company. Within two years there were four or five people from that group who had started their own companies. It takes one fool to do it.

Johnson: And then everybody says, "Wow."

Kolence: Yes.

On the measurement stuff, I was so careful to do all the experiments and get everything together. Now we're dealing from 1958 to 1967. It was at least 1968 or 1969 before we had the products. Eleven years.

And I was just working away on it the best I could. I was real slow, but the thing was by the time that the 360 came out, that's when there was really a need for it. People were so dependent upon performance that they had to have it.

Johnson: Everybody gives IBM the credit for starting the software industry when they unbundled. I think the fact that they weren't able to deliver software for that 360 is really what started the independent software industry.

Kolence: Yes.

Johnson: They went out and sold all those machines and there wasn't any software that would work on them and that gave people an opportunity to step in and fill that need.

Kolence: I agree with what you're saying but, in addition, you had a stable line of machines that were plug-compatible. You could put the money into developing a product and even if they came up with a new machine you could afford to port it over there. I think that had a lot to do with it too. IBM stabilized the industry with plug-compatible machines.

Founding of Boole & Babbage

Well, that only brings me up to the Boole & Babbage days. After that, when I did the Institute for Software Engineering, that's when the idea that you could manage performance, equipment planning, capacity planning, performance management and all that stuff became an accepted part of every major installation. Up until that time it was something the systems programmers did.

When we started Boole & Babbage, we knew a lot about development and really were interested in doing the sampling monitor. We also did an equipment configuration monitor – equipment utilization. It was the same thing.

The critical thing was that nobody could figure out that you didn't need to sample real fast. Everybody thought you needed to sample real, real fast because the computer runs so fast.

Johnson: They didn't see the sample as a snapshot in time, so to speak?

Kolence: No. They didn't really understand that it was population samples. If anybody knew any math, they thought it was signal pattern sampling and not too many people knew about that in those days.

At any rate, Dave wanted to go in business for himself and I had never thought of it like that exactly but I didn't want to leave Palo Alto. We felt we could really do a good job of helping people if we demonstrated the design processes. And I had a lot more ideas than just what I had done.

We started Kolence and Katch or Katch and Kolence in May of 1967. And by October, we were doing consulting. What we were going to do is get a consulting business going and then as we had some money put together we were going to build the measurement products.

So it always was intended to be two things – the design and instrumentation and the engineering side of things. Let's go back a little bit and talk about engineering.

In 1962 or so I went off to Edwards Air Force Base from North American Aviation. And there I saw an airplane take off for the very first time. As it was taking off, I said to myself, "Wow, that guy is trusting his life to the engineers. And I'd never trust my life to software. So we need is an engineering discipline."

I invented the term independently, okay? Because there were other people who also had the idea at the time of software engineering. I started calling myself a software engineer. And when we founded Boole & Babbage, we didn't call the people programmers. We called them software engineers. And that's where the term software engineers came from in Silicon Valley.

We were going to do consulting, make some money and get a little backup so we could develop the products. And then finally we got some venture capitalists that would listen to us. There were three of them, or maybe four. But the lead guy was Franklin Pitcher Johnson.

He was Franklin Pitcher Johnson, Jr. so his dad had the same name and was a track coach at Stanford. So they invested in Boole & Babbage. We had to put up $5,000 apiece. Now, that was a lot of money in those days. And they put up $50,000 and guaranteed us a loan of $100,000 if we needed it. And for that they got 90% of the company. So that's how Boole & Babbage got founded. Nobody else would touch us because software, what's that? But Pitch actually knew Fortran.

Johnson: Oh, he did?

Kolence: Yes. He knew Fortran and he had worked in the steel industry as a foreman. He had an MBA. He's only a couple years older than I am. But in 1966 there weren't any software companies and that's how we got started.

And then we lost our main consulting contract which was with Fairchild Semiconductor because they fell on hard times and cut back on their consulting a lot. So we took the $100,000 and we built the software products that we wanted to build.

Johnson: How many people did you bring in to do that?

Kolence: We only had about five or six people that were programming. Later on we brought a couple people aboard to do marketing.

Johnson: Yes, but you were talking about building the products.

Kolence: Five or six guys. I didn't program it because I was the guy writing up the marketing material and talking about it. I hadn't programmed the 360 at all. The 360 was a brand new, wasn't working-well, didn't have multiprogramming.

MFT wasn't even out. It wasn't until Release 13 that it started to become stable. The releases were coming out pretty fast and we hired a guy that knew the guts pretty well and another guy that was a sharp programmer and learned it fast.

The patent we eventually got on the sampling monitor was in my name and Gary's name. I can't remember his last name. We later on lost the patent because I had published an ad in *Datamation* more than two years before we filed. The *Datamation* ad said nothing more than some little stuff about performance.

I should also point out that at that time people weren't selling products. They were selling software. And we had this problem of selling something and then keeping it up-to-date with the latest releases of all the operating systems. So I invented the annual maintenance charge.

We were the first software product house in Silicon Valley. I don't think there were other people who were building software products whether they called them that or not in Silicon Valley at that time.

Johnson: I don't think so at that time.

Kolence: There were one or two down in Los Angeles and Marty Goetz [of ADR] was doing it with a sort program in New Jersey.

Johnson: Both Informatics and ADR started out with programming services. And they got into the products business because they developed products for the clients.

What was the name of the initial product you developed?

Kolence: PPE. Problem Program Evaluator. The reason it was called Problem Program Evaluator is we crippled the ability of it to evaluate compilers and IBM systems software because we didn't want to earn their wrath.

Johnson: Okay. So in other words you were really focusing on the programs written by the application programmers in-house. Interesting.

Kolence: Yes. We crippled it on purpose. I specifically said that we didn't want to get in harm's way.

Johnson: Right. And then the other product was?

Kolence: CUE. Configuration Utilization Evaluator. I never was any good at names except Boole & Babbage wasn't a bad name.

Johnson: No. That was a good name.

Kolence: That's what we started with. Because they were basically the same engine. We were looking at different things every time we had an interrupt.

After that got going, the next thing we built was a dataset optimizer where we would look at where a dataset was placed on a disk and determine how much time it took to seek it. We didn't use linear programming because it's too gigantic of a problem but we used an iterative method to do a local optimization which was not bad. Given that you had nothing before that.

CUE got us in a lot of trouble though. The reason was that people bought 360s and they added channels and disks and stuff like that and of course the performance was horrible. So the salesmen were selling faster and faster CPUs and not upgrading the I/O. When you ran CUE, you instantly found out where the bottlenecks were.

Remember that IBM used to have levels of account control. Level I was you do all the equipment planning for the account. Literally. And this threatened account control in a big way.

And it was because we had this systems analysis software, which was cheap, too. We sold them for $5,000 apiece or $7,500 for two.

<u>Marketing PPE and CUE</u>

So CUE was a big success and PPE was a big success, too. Our marketing strategy was that we'd say, "Okay, sign a contract and we'll come out and you give us three programs to run and we'll run PPE on them. Give us 24 hours and we'll give you the corrections and rerun. If we haven't saved you at least 20 percent in your runtime, the demo is free." We got every one of them. Signed them up. We were saving them 30, 50 percent.

Johnson: Were you doing this locally?

Kolence: We were going all over the country.

Johnson: Were you advertising in *Datamation*, *Computerworld* and publications like that?

Kolence: Yes. We had really good press from *Computerworld*. We didn't have to do much advertising.

Johnson: So they were calling you up and saying we heard about your demo and asking you to come out and show it to them?

Kolence: Plus we got the word out through SHARE. And we had some salesmen out there going to the big sites.

So we were selling pretty well. But then IBM decided that they were going to write the same program after they got a briefing from us at SLAC, Stanford Linear Accelerator Center.

They had a couple of their software engineers write the same programs and put them into NASA COSMIC for distribution. And everybody was suddenly taking those and not buying them from us because they were only $500 or something like that.

Johnson: Oh, my goodness.

Kolence: So we went from making money to losing lots of it.

Johnson: When did this happen?

Kolence: That was probably in the spring of 1970.

Johnson: I never heard that story.

Kolence: Oh, yes, they really did us in. I went over and talked to Paul Armer at SLAC. Begged him. I said, "Look, we invented this. Please don't release this." And he looked at me and he said, "You know, you should have done that in hardware."

I saw red. Literally. And I couldn't do anything except spin on my toes and walk out.

Johnson: So you had a couple of good years with this product before IBM came out with theirs?

Kolence: About a year and a half, actually, after we got it out. We sold and delivered the first product in January or February of 1969. We delivered it to Guelth University outside of Toronto.

My marketing strategy was to go out and show people how it worked. The word got around fast but people didn't have any money. There was no such thing as a budget for software. So I was told on several occasions that the choice was between buying my software or buying another 2314 or 2311 disk drive.

But we were starting to do okay. Quite frankly, one thing that happened is I started to get too full of myself. You know how that can happen. I thought I was the only one who knew what the hell we're were doing and so I turned out to be a horrible president. That's okay. You learn from that.

I resigned every month for six months. And at the end of the six months I got fired by Pitch. That's exactly what happened. But that's the way it goes.

Founding of the Institute for Software Engineering

After that, I became an independent consultant and I wrote a book called *Software Physics* which has to do with the hundreds of different metrics. You have underlying, independent variables. How do you use them for workload forecasting? How do you use them to predict? How do you use them for equipment planning, capacity management? I defined what that was. And then I started another company called Institute for Software Engineering because I wanted to bring in the design stuff.

But then I turned 50 and I didn't want to keep running the company any more and I sold it to Boole & worked there until I quit in '86. I hated it although I did some really interesting work. I was in charge of consulting.

One of the things we did at the Institute for Software Engineering was we taught all this stuff. We didn't do just consulting. We also ran competitive analysis experiments for IBM , DS-Sort versus Syncsort versus VA-Sort. We ran those every so often to let them know how they compared. And also for a while there we tested other people's hardware. Hitachi versus IBM.

The thing that was really interesting, I finally convinced them to let me try and calibrate the performance of things like VA-Sort in terms of the workload characteristics and that sort of thing.

We came up with absolutely beautiful engineering-type curve that showed that there were ten microseconds difference between the same record sizes. There was an increase of so much on the keyword that by volume it flattened out to average. Did a beautiful job on that. That's really what I want to do now for the objects-oriented world is build engineering characteristics for software. Software generic characteristics.

Johnson: Let's see how we did on the questions I had.

Kolence: Okay.

Johnson: Just to clarify, I gather that your primary competition was the free programs from IBM.

Kolence: Well, our main competition was that people didn't believe in buying software.

Johnson: Yes, that's consistent with what I'm hearing from other people who were out there at the time.

I think that wraps it up. Thanks so much for your time, Ken. This has really been interesting.

Kolence: My pleasure.