Oral History of Norman Nie

Interviewed by:
Luanne Johnson

Recorded: June 24, 1986
Via telephone

CHM Reference number: X5545.2010

Table of Contents

Norman Nie

Conducted by Luanne Johnson

Abstract: Norman Nie discusses the development of the SPSS statistical analysis software product was developed when he was a graduate student at Stanford University and how it was distributed by Stanford and then the University of Chicago until the IRS determined that the income received from had to be declared as unrelated business income which forced Nie and his associates to form the company SPSS to distribute it as a commercial product. He describes how this resulted in a model for commercializing products developed in university research laboratories which has become widely used in many industries and how the funding of the development of SPSS differed from similar products developed by early competitors. He presents his view of IBM's role in forming the early software industry and his view on how the difficulty of developing a second successful software product is a significant inhibitor to the growth of the software industry.

Development of the SPSS Product

Norman Nie: SPSS is one of the few honest-to-God bootstrapped companies out there as opposed to some others in our corner of the industry. A number of the ones that you mentioned in your letter really did have early capitalization of a substantial sort.

In 1965, I was a second year graduate student at Stanford University in political science. I was a research assistant at the Institute for Political Studies at Stanford and they asked me if I would help bring them into the modern computer era. I was one of the few people who had had some experience in programming computers and doing data analysis on something other than old-style counter/sorter tabulation equipment.

I was appointed Director of the Data Analysis Laboratory as part of my duties and my research assistanceship. I went around and gathered all of the then existing statistical analysis programs, such as BMD which ran on the [IBM] 7040 and the [IBM] 7090. At that point the university had a Burroughs 5500 and an IBM 7090. The problem with the Burroughs 5500 is there was no software around. Interesting machine though.

I spent a year trying to pull together things from IBM's Share library and BMD and became extremely frustrated. The primitive level of this stuff in 1965 was really just unbelievable. So I decided that I wanted to design a system and I had a buddy who was getting his Ph.D. in Operations Research. He was in about the same stage of graduate school that I was. His name was Dale Bent and I conned him into starting this project.

There were two motivations. One of which was the job in this lab and the other was the research project which I was working on with a chap who's now at Harvard and out of which my dissertation was to come. I knew that there was just no way we could handle that mountain of data with the then available tools. So I began filling notebooks with the designs for SPSS.

To make a very long story short, a year later I found Tex Hull and now we really had two sophisticated designers and a super programmer. Within a year we converted, and mostly text converted in terms of actual coding, these two giant black notebooks into the first release of SPSS.

Luanne Johnson: What year was that?

Nie: That would be 1967. When we released the system, several of the people were so impressed with it at Stanford that my mentor, the chap I was studying with, said, “You've really got to get this out to other people.” He was one of the leading people in the country in quantitative social research. That's Sid Verba, who's now dean at Harvard. He sent out a letter to about 200 social scientists around the country saying that they really ought to take a look at this thing and that it was a major innovation. He and I sat down and decided that we had to charge something for it because there were costs incurred as people ordered it so we came up with the original pricing. Which was basically what a professor could go to his chairman and get without having to bother to go to a dean. That sum was probably $500 or less and we knew that sounded too round, so we decided it was $400. That was how the original pricing was established.

Johnson: At this point did the package belonged to Stanford?

Nie: No, it belonged to me. It was a research project. It belonged to the three chaps who were doing it.

Johnson: Okay.

Nie: We had no institutional funding of any substantial sort. And this is back in the period when universities hadn't figured out who owns software when it's developed by their faculty. Universities *today* haven't figured that out. Because books are owned by professors. And research projects are certainly owned by the faculty as they move from one institution to another. It goes with them. It's the notion of the principal investigator. It really is only patents that have run in the opposite direction.

Johnson: Well, the reason I ask the question was that there was quite a lot of software out there in the late 1960s. I researched the directories at ICP for software that was being offered during that time. The BDM products were being offered out of UCLA.

Nie: But you see that wasn't owned by UCLA. And that's a private company now. That, however, is extremely different because they had $20 million over a ten-year period of NIH [National Institutes of Health] funding.

Johnson: Yes, that's correct and they weren't in a position to market it because the government funding meant that the software was in the public domain. The same thing with University of Michigan which had a number of software products but couldn't really market them. But that's what I wanted to clarify. This was your research project that belonged to you which is different than the other situations.

Formation of the Company SPSS

Nie: By 1968 there must have been 150 copies of the SPSS software out there at various colleges and universities. I took a job at the University of Chicago as Assistant Professor. Dale Bent went off to the Business School at the University of Alberta. And Tex Hull disappeared into the Valley to work for a commercial company in the hardware business.

I took the package with me to the University of Chicago, basically doing nothing with it other than having an ordering system whereby people could order it over the next four years. By 1974, roughly, the product was generating a couple hundred thousand dollars a year of revenue. Sometime in 1972 Tex Hull came to the University of Chicago to accept a position I had nominated him for as Assistant Director of the Computing Facility.

And somewhere around 1974 the IRS audited this research institute where we housed SPSS and said, "This is something that ought to be taken out of here. This is not part of your charter. This is unrelated business income. So get it out or declare the unrelated business income." And that's something universities do not want to get involved in. The University said, "Norman, this is yours. Get it out of the University."

So that's what I did. We fussed around awhile trying to get our own 501(c) status because I had no desire, training, background, or inclination to go into business. I was a full-time tenured faculty member of the University by that time and had met my wife Cora. I had some idea that there was some money to be made off the software. And certainly off the textbook that I had written to accompany it because that was making lots of money selling 60,000 – 70,000 copies a year at that point in time.

But anyway, we were turned down for 501(c) status and I decided to set it up as an Illinois corporation. And over the next ten, eleven years, it's grown from that to 220 staff members and a $25 million a year company.

So the original origin was a group of guys, one of whom (me) knew what data analysts want who aren't professionally-trained statisticians. The real strength was the ease of command language and self-explanation of the output. It's the key concepts that I had about self-documenting files and so on. And we had a pretty decent statistician, Dale Bent. And Tex Hull who was one of these 1960s absolutely super coders. To this day he's still with the company as a vice president and to this day he can still crank out more bullet-proof code in a year than six other programmers.

Johnson: During that period of time when you said that you had a system so that people could order it, obviously there was quite a bit of demand for it. I presume that this was primarily word-of-mouth generated demand.

Nie: Yes, reputation, word-of-mouth and the book which we published with McGraw Hill. It had set a new plateau for what documentation for software ought to look like.

Johnson: That was the textbook you had written to accompany it.

Nie: Yes. And that's because I was a professional academic who taught research methods and I looked at it as a tool to teach that method of research.

Transition to a Fully-supported Software Product

Johnson: To what degree did you have to provide some sort of support service? Did people who had the software and the textbook feel that they needed to call you to ask questions?

Nie: Amazingly very little. The first few years we had almost no help at all. It was such a quantum leap in capability and ease of use over what people had been dealing with and the documentation was so good and the system was still, at that point in time, relatively simple. I mean, you're talking about, at that point in time, a 40,000 line Fortran program. The demands for service were minimal.

Johnson: Did that change? The reason I'm asking these questions is that it's very clear that a software product is more than just code. There were some experimental companies in the late 1960s that set out to broker code. They were going to take code that had been developed for a particular user and resell it and so the company that developed it could recapture their investment. That was just a complete failure.

Nie: Absolutely.

Johnson: The companies that have been successful in most cases define their software offering as consisting of more than just the code. In your case you had the textbook. Other companies also provide documentation but also maintenance service, enhancements. and so on. I'd like to understand to what degree you had that initially and when you began to develop that as part of the product. As I understand it, now you do have a typical fully-supported software product offering which includes maintenance and enhancement. Is that correct?

Nie: That's true.

Johnson: I'd like to talk about the progression from just being a program and an accompanying textbook to becoming a fully-supported software product.

Nie: Okay. Somewhere 1973, 1974, we started doing enhancements to the product and fixing accumulated bugs. I think probably 1972 or 1973 was the first new tape that was sent out to existing customers. And then from about 1974 on, we began getting serious about added routines, enhanced quality, cleaning up conceptual problems in the system, etc.

But what drove this was that it was a unique product. Look, BMD existed before us. They had already had millions of dollars put into their software by the mid-1960s. But there was no system concept. It was a set of standalone programs and the only thing they had in common is they all began with the word PROBLM, okay? And then you had a one and a six and a yes and a twenty-three. It was like reading external formats for generalized use but there was no language. There was no way to understand it.

Well, we came in with an English language-oriented system. Here is my list of variables. Here are the values that each of them may take. Here are labels for each of the variables. Here are labels, if you want them, for each of the values of those variables that have discreet categories. This variable is an alpha-type variable. This variable is an extended string. This variable is a numeric variable with the following characteristics. This literally read like English. Now regress A on B.

Second, once you defined a file like that, the next thing you did is save your file and it stored all that in the internal machine-readable form. So the next time you got it you said, "Just get my file. And now regress." Education, occupation, income, on direction of presidential vote. It was that self-contained.

There were about 15 statistical procedures, each one of them described in great detail in terms of not only how to use it but what it's good for and really pedagogical material. And it supported itself. It did so really until we started getting significant competitors. The drive to add new facilities and update the architecture, that's a 1976-onward phenomenon.

Johnson: So it was competition that began driving you to think in terms of ongoing enhancement and upgrading.

Nie: That's right. Protecting our market share.

Johnson: Who were those competitors that began coming in?

Nie: There was really only one significant competitor, SAS in Raleigh Durham. They are another one of these groups that had very substantial amounts of federal and state government funding.

Johnson: Have you ever had any outside funding?

Nie: Nope. Well, I think we had about $6,000 or $7,000 in a computer time research grant through the Institute at Stanford. And the fact that all of us were working on advanced degrees and, in one way or another, employees of Stanford as graduate students. Other than that no financial investment of the package at all. The original stock in the company was $500 each.

Lack of IBM Competition

Johnson: That's a great story. When you're saying that you didn't really have any significant competition until SAS began to show up in the marketplace in the 1970s, was IBM offering a free software package that could be considered competitive?

Nie: Yes. But it was such a piece of junk that nobody would take it.

Johnson: This history project started to try to determine the impact of IBM free software on the formation of the industry. And I'm finding that with applications software what IBM had to offer was usually so poor that it didn't really provide any significant competition.

Nie: That's exactly the case. Once in 1976 IBM invited us out to Armonk to talk to us about our product. And their starting line was, “Didn't we invent this?” And we had to tell them that what they had wasn’t at all comparable.

Johnson: You have a unique perspective in the sense that you weren't really oriented around running it as a business until you were forced to.

Nie: That's right. We created the product which was never viewed as anything but a scientific research tool. Which then, due to its own success, began to ring our bell. And if you really want to know the critical issue, the IRS put us in the business.

But the model that we provided of an academic operation which then gets spun off into private ownership has over and over again been used. And I think we were probably the first. All of our competitors in our field followed us down that road, BMD at UCLA, SAS at North Caroline, Minitabs at University of Michigan. You can just go right down the list. And then the whole group that starting selling line-oriented text editors out of the WYLBUR group which was invented at Stanford at the same time we were there. You just go right down the list and you see this phenomenon occur over and over again.

Johnson: That's interesting. That's, of course, a model that's being used in all kinds of fields now. Was that really a unique model at the time that you did it?

Three Models for Forming Software Companies Prior to 1972

Nie: I think we were probably the first to do it.

There are in my experience, with enterprises that got off the ground prior to 1972, really only three models. One of which is that you develop it within an academic institution which provided you with the time. That's what the institution really provided us. We got no support. But there were three guys who were being paid as research assistants or had fellowships who had the time to develop the product. That's model number one.

Model number two is you get a customer who you are doing specialized programming for to pay for your development. Clearly, ISSCO followed that model. Peter _____ got large key customers like CDC to actually pay for the generation of what then turned out to be general programs.

The third model is that the federal government or some level of government really paid for it. Up to the early 1970s, it was the only way to capitalize software development.

Johnson: What I'm trying to zero in on is what elements were affecting the way the industry developed in the late 1960s, early 1970s. The people that I've talked to were trying to establish a business in those days. You're in a unique position in that you understood what a software product was but you weren't at that point actively trying to create a company.

Some people, particularly those who were selling system software, saw that IBM was a significant factor because they really had a difficult time selling against IBM. Some people felt that it was really much more a matter of educating the customer as to what to expect from a software package and that it was possible to buy a prewritten program that would work. Some people see that the real problem that the industry had getting off the ground was financial and human resources that had to be pulled together that weren't there.

Starting Point for the Software Products Industry

Nie: You can't have spent your life in this business without giving a lot of thought to those questions. Let me see if I can summarize what I think. I think the following things happened. One, we were in an era when the compatibility issues were driving everybody crazy. Because the utilization of computers had begun to take off. And every two to three years a new machine was brought in. Even within a computer manufacturer's product lines there was no upwards compatibility. It is true that a program written on the 7090 or 7040 or their business equivalents might run on a 360, but, boy oh boy, it didn't take any advantage of why you bought the new box.

And when you went across manufacturers' lines, forget it. We were just emerging out of the 1401 mentality with limited machine size so that programming in source languages was rare. It

was good in concept but you couldn't get anything to run with any efficiency or any size. You were dealing with very little capability.

So IBM's introduction of the 360 was really the starting point. It was a very, very capable machine which failed to deliver a number of things that it said it was going to deliver down the pike like interactivity and so on. But it was a capable machine that took some of the toughest screws off with enormous popularity across science, industry, academia. And yet it required another substantial investment in software. At that point in time I think people began to see that you could write stuff for what was clearly going to be a huge installed base. And it may be cheaper to acquire than to write your own.

You see, general purpose programs were there all along. You take the area of statistics. I did my undergraduate work at Washington University in St. Louis. We had an IBM 740. There was a series of programs available. There was a regression program. There was a cross-tabulation program. There was a factor analytic program. Each of these was written so that with minimum of patching and fussing and so on someone could bring a new data set to it.

The concept of a generalized program which could be reused by a number of different users was already there by 1964. By 1962, actually. And what happened was people took the next logical step forward. People began to see that you could package a number of these things together and that people would find it very attractive and be willing to pay for it and that there was an enormous cost advantage over doing the same repetitive task.

When I was the statistical consultant at Stanford, the head of this laboratory, we had two full-time programmers who did nothing but write quick and dirty programs. And at the end of a year in looking at what these chaps had done, they'd written mostly the same program from scratch over again ten times. Then you look carefully and they didn't write it from scratch. They just started stealing things from their prior programs. So it was a very natural step to say, “Hey, let's take a systems approach to this.”

And from the user's perspective, you saw the beginning of the professional DP operation where someone was responsible for a service center. And people began to start saying, “What kind of tools do we need to run the service center?” Well, everybody wants efficient sorting and huge numbers of these shops went out and built their own. You know, everybody threw out IBM's sorting facilities. Half the major corporations in America built their own.

Johnson: Oh, really?

Nie: Oh, yes.

Johnson: I was around in those days but I wasn't aware that there were that many that were building their own sorting systems. That's very interesting.

Nie: If you had hundreds of thousands of records you weren't going to use the sorts coming from IBM. You just bought this big new machine and you might as well forget it and buy a second one. So you had competitors to IBM like Syncsort. People started looking at methodologies to do these things. Some of the earliest software companies were systems utilities companies. Well, that stuff hardly exists anymore because IBM really *has* driven almost everybody out of it.

Johnson: You're right. Another type of product that was very popular was spoolers.

Nie: Yes, spoolers. And the generation just before that non-hardware vendors provided assemblers. We must have had six on the 7040s and the 7090s, the MABs and the FAPs and so on that were non-IBM produced.

The biggest constraints were we didn't have any idea of how you did project management and coordination of a big program. I'm the first person to admit that all my designs and great ideas and natural language wouldn't be worth crap if I hadn't accidentally stumbled on a programmer of the giant capability of Tex Hull. Because all the notions of top down programming just didn't happen.

And then I think finally IBM's failure to ever be able to produce good applications software are led people to look elsewhere.

Johnson: So in many ways IBM really was a very influential factor in the market by virtue of the fact that they were producing such poor software products. They had more influence in creating the software industry by the fact that their products were poor than the fact that their products were free.

Nie: That's right. And on top of that the fact that they had an enormous installed base so people figured out that they could spread the cost of developing the products over a large number of machines and make a living doing it.

Comparison of the Marketing Stategies of SPSS and SAS

Johnson: Did you find with the type of customers you had initially – the universities and research laboratories – were preconditioned to expect to be able to use a generalized purpose

package? Did you have a lot of resistance from customers that said that it wouldn't work for them because their situations were unique?

Nie: No, not at all. Realize that this is an field where you study statistics and if you then go into the general business world you're really an ex-academic. That's what drove this market. Master's degree students and Ph.D.'s and ABDs got put into government research agencies and in businesses, etc. So you weren't selling to the DP marketplace, you were selling to an end user.

Johnson: Right. Most software vendors began to do that in the late 1970s when they began to bypass the DP manager and go to the end user. But you were doing that way back in the very beginning.

Nie: In fact, the biggest distinction in the business between us and our main competitor is that SAS is an IBM product, back-to-back and belly-to-belly in design with IBM machines. And they sell to the DP people. Because they've designed the product to look like a PL1 program. And we're an end user program. We don't appeal to the DP manager.

Johnson: Interesting distinction. And certainly very unique at the time that you were doing it. Because it dawned on the rest of the software vendors much later that the product should be directed towards the people who were really using them as opposed to the intermediary.

Nie: Well, it depends on who has the purse strings. You know, what's really broken the back of that has been the PCs.

Johnson: Sure. Have you dealt with PCs at all?

Nie: Oh, yes. We have *the* leading statistical software on the PC. We dominate that marketplace.

Johnson: Oh, you do?

Nie: Yes, that's where our real growth is now.

Johnson: Great.

Nie: We have over 20,000 units out there.

Johnson: That's terrific.

Nie: And they're expensive. They're $800 a piece plus an additional $1,000 in option you can add on.

Act II Problem as a Growth Inhibitor for the Software Industry

Johnson: Great. Well, I've gone through the list of questions I had. Is there anything else that you'd like to throw in, any particular ideas that you have about the growth of the software products industry that you'd like to talk about?

Nie: Well, I think, the entire industry is a very funny one in the sense that so many of the companies thrive for a while and wither because of the Act II problem. What I mean by the Act II problem is, let's take us, for example. A program like SPSS got started because you took a guy (me) who really, really knew what a particular application was for an end user. I knew exactly what he needed. Plus we had a system architect and a super coder. You put those three together and it was a unique set of talents.

Now the question is: what do you do next? Because then what happens is you've got a business searching for markets and you no longer have that intimate relationship between the small team of people who built it and a natural market which they are experts at. They're no longer experts, because they've been out of it for 10 or 15 years.

I think that produces a real Act II problem. If you look at Lotus, they're clearly flailing around to figure out what to do next. They put all that money into Symphony. And it turns out that's a very neat way to look at spread sheets. It's not a particularly neat idea for how to do word processing or graphics. Therefore, just simply putting 20 programmers in a room and grafting applications onto your approach is not always the winning ticket. And I think if I look throughout the industry this is happening again and again.

Johnson: Yes, there have been very few companies that I can identify that have been successful with add-on products. There are some examples in the accounting applications area where a company like MSA starts out with an accounting product and then adds on other accounting products.

Nie: Well, once you've been really successful the question is where do you get your second revolutionary product? And your third and your fourth so that you really are protected both in time and from competition?

And that, I think, the industry hasn't even begun to face yet. With the exception maybe of Microsoft which got into a number of very different areas, compilers and applications and windowing tools and so forth. With perhaps Microsoft being the real exception, most of this industry – even the most salient players in the top 50 – are really caught in this problem. It isn't like hardware where the revolutionary designs seem to follow from each other because of the intimate human behavior aspect of software.

Johnson: Have you successfully brought out a new different product?

Nie: Nope. We've got some partial successes and a couple of failures. We're much more elaborate, much more capable, ten times the size and twenty times the facility. But the basic system and language is still the same.

But as we move into micros, that language and system isn't totally applicable for full screen. You can tack on menus and so forth. But we need to have a new architectural start. The problem is the systems are so capable that now you can't sell and distribute the kernel that we did in 1967. Now, you're talking about a $15 million investment.

Johnson: Sure.

Nie: It's very interesting. Kind of building your own insurmountable wall.

Johnson: Right. I think following this industry over the next decade is going to be just fascinating.

Nie: I think you've got a real interesting problem.

Johnson: Well, I'm just starting with the idea of going back and tracking what happened in the early days because I can't find anybody who really has pulled that information together and observed what was going on.

Nie: I think the picture looks very different depending upon what kind of software. The system utilities are very vigorous. Syncsort's still a very alive and well company. Databases are obviously the most high ticket one. Statistical software, accounting software, payroll and warehousing. I think you have to go through each of these areas because I think they've all got different mechanisms, funding and different histories.

Johnson: Yes, they do. And overall it's still such a small industry compared to other industries that it's thought of as a single industry.

Nie: Yes, that’s true. Sorry, I've got to run. I have another appointment.

Johnson: Okay. Thank you so much for your time.

