Oral History of Harry M. Markowitz

Interviewed by:
Luanne Johnson

Recorded: May 14, 1986
New York, New York

CHM Reference number: X4804.2008

Table of Contents

DEVELOPMENT OF SIMSCRIPT AT RAND CORPORATION 3

FORMING CACI 5

UPGRADES TO SIMSCRIPT 6

COMPETITIVE PRODUCTS TO SIMSCRIPT 7

PRICING ISSUES 8

DATABASE LANGUAGE PLANNED FOR SIMSCRIPT 11

DEVELOPMENT OF QWICK QWERY 12

DEVELOPMENT OF EAS-E AT IBM 13

EVOLUTION OF REUSABLE CODE TO SOFTWARE PRODUCTSS 14

Harry M. Markowitz

Conducted by Luanne Johnson

Abstract: Harry Markowitz describes the development of the Simscript simulation language at Rand Corporation and how he and Herb Kerr founded CACI in order to market Simscript I.5 commercially. He discusses pricing issues and competitive products to Simscript and to CACI's data storage and retrieval product, Qwick Qwery. He covers the evolution of the Simscript product from the initial version developed at Rand and the acquisition by CACI of Simscript II Plus, developed by Phil Kiviat, which became the commercially successful product Simscript II.5.

Development of Simscript at Rand Corporation

Luanne Johnson: I'm doing research into the beginnings of the software industry. I have a background in the software industry myself. I founded a company in 1971 and sold it to the employees a couple of years ago.

Harry M. Markowitz: Okay, good.

Johnson: Since then I've been consulting and doing committee work with ADAPSO. Burt Grad asked me to work with him on an article for *Datamation* about the impact of IBM's unbundling. I think you probably know he was on the IBM unbundling task force at the time.

Markowitz: Right.

Johnson: In the process of doing that, I realized that nobody has really done any research or collected any information about the early software products industry. A lot of people are writing about the microcomputer software industry and a lot of those company founders have become folk heroes. But nobody's really looked at what happened in the early days of the

software product industry in the late 1960s. So I decided that I want to do that. This year is ADAPSO's 25th anniversary and I'm contributing material toward that.

Anyway, Burt told me that you were involved with CACI at the time that they were marketing Simscript.

Markowitz: Right. Another fellow, Herb Kerr, and I founded CACI. Shall I give you a brief history of how Simscript came about and then how CACI came about?

Johnson: Yes, please tell me how you got into that.

Markowitz: I was at the Rand Corporation in the 1950s, starting in 1952. Rand was doing simulation at the time and I was involved in that, particularly in the logistics department. I helped design and develop a man-machine simulation for the logistics department and did some other logistics simulations. The thing that was characteristic of simulation in those days was it took an awfully long time to explain to the computer how to be an Air Force base or manufacturing facility. I had a theory that what was needed were reusable subroutine packages that you could put together in different applications.

Johnson: This was when?

Markowitz: In the late 1950s. My memory is very bad on dates so every date is plus or minus a couple of years. But somewhere around 1959, I got an offer from the General Electric Company Computer Department. I went there and later moved to General Electric Manufacturing Services, one of the services departments at corporate headquarters. In cooperation with a one of the GE departments, we built something we called the General Electric Manufacturing Simulator, GEMS. This carried out the idea that we'd make packages of reusable subroutines that could increase flexibility. It would take us a little longer in the first instance but it would be faster to build things in the second instance. The particular applications that we developed this for were job shops. General Electric had lots of job shops.

It turned out that these reusable subroutines weren't all that reusable and flexible, except for a few basic ones that had to do with things like creating entities, packing information into entities, filing things into sets, removing things from sets, causing events. These were very basic. These were reusable. So my next theory was that what we needed a language that would allow you to create, destroy, file, remove, cause events and so on.

I looked for a nice Rand-like environment in which to develop this idea. Finally, I ended up back at the Rand Corporation. I had the good fortune to have assigned to me Bernard Hausner as my programmer to implement this idea. We first called it SPS1 (Simulation Programming System One).

Bernie and I put together a first implementation of the SPS1 in about a year. Then an old friend of mine, Herb Karr, had a falling out with Planning Research Corporation and was at loose ends. We hired him as a consultant to write the SPS1 manual. In the second year of SPS1's history, we planned revisions of the SPS1 language on the basis of experience with the initial version. Herb wrote a manual, not for the existing SPS1, but for what the product was to become. We also chose a new name for the language, out of various candidates, by putting SIM and SCRIPT together.

So Simscript was a product of the Rand Corporation. It was used at Rand fairly quickly and extensively, primarily for logistics simulations.

Forming CACI

Over the years, Herb Karr had suggested that he and I go into business together and I had said no. But when we were done building Simscript it seemed like a good idea so I said, "OK, let's go into business together." We formed CACI early in 1963 or late in 1962. It was just him and me. I remained on as a consultant to the Rand Corporation. He was a consultant, I think, at Douglas [Aircraft]. We had one or two consulting contracts as part of CACI. And we gave our first Simscript course. At first we each put a thousand bucks into the business. Later we had to put in another five thousand bucks. Much later we went public.

Our first Simscript courses were reasonably well-attended. We got roughly 20 to 25 people to come to each course for a few courses per year. We thus started to bootstrap CACI.

Johnson: Was Rand selling Simscript and you were training people how to use it? Or did you take over marketing on it?

Markowitz: Rand had put Simscript into SHARE. It was in the public domain. Rand is a non-profit organization.

Johnson: Okay, I understand. Putting it into SHARE was a very common thing to do at that point.

Markowitz: Right. So it was in the public domain. We weren't building or maintaining it. We were teaching it in the first instance, and offering to apply it. One of our first contracts was with Navy Logistics in Mechanicsburg (Pennsylvania). We got a contract for one person full-time. So we hired one person and sent him off to Mechanicsburg.

Upgrades to Simscript

So we continued bootstrapping. We got a contract from IBM to modify Simscript so that it would work on a new operating system that was just coming out. While we were doing it we rebuilt Simscript.

Johnson: Do you remember if that was with the System/360? It was about that time.

Markowitz: That's plausible. I don't know whether it was for the 360 OS or whether the 709 had an operating system distinct from what we had built it for.

We did this for IBM's Los Angeles Scientific Center. While we were at it, we changed Simscript in a couple of ways. In particular, we got rid of certain kinds of restrictions in the original Simscript. For example, Simscript has what we call control phrases such as "For I = 1 to N" or "For each job in Queue of Machine." These could be combined with phrases that instructed the computer to only process items WITH this or UNTIL that. Current Simscript allows the programmer to form any sensible combination of such control phrases and use the combination to control one or more action statements. In the initial Simscript you were not permitted to use both "For each of set" phrases and "For I = 1 to N" phrases in the same combination of control phrases.

We removed that restriction and generally upgraded Simscript I to Simscript I.5. As far as the language was concerned, it was a matter of removing a few artificial requirements. We also rebuilt the internals of the compiler. Instead of putting inserts into FORTRAN and passing the whole thing to the FORTRAN compiler, we developed a language with which we could describe the syntax in a table-driven way. We generated assembler code which was then handed to the assembler. .

So we at CACI rebuilt Simscript (1) and called it Simscript I.5. I was still a consultant to the Rand Corporation where we had started to work on Simscript II which had a lot more features than Simscript I. I mean, Simscript II was a whole new language. It was a general purpose language with a simulation capability. When CACI got the contract to upgrade Simscript I, we rebuilt it using the kind of language generation technology that I had developed at RAND to implement Simscript II. Thus Simscript I.5, CACI's proprietary product, was Simscript (1) with

some restrictions removed, implemented in the manner that Simscript II was being implemented.

Johnson: I see.

Competitive Products to Simscript

Markowitz: Since nobody was maintaining Simscript I out of SHARE, Simscript I.5 became the standard commercial Simscript. There were several simulation languages that came out at the same time of which GASP is still going. GPSS was, I guess, the most popular at the time.

Johnson: Yes, that one I've heard of. I hadn't heard of the other one.

Markowitz: Right. GASP was, I think, originally designed by Phil Kiviat. I think GASP IV or GASP V is still going under Alan Pritsker. Anyway, so Simscript I.5 became our standard product.

Johnson: So at this point somebody who wanted to use a simulation language, could still have gotten Simscript I from SHARE.

Markowitz: Right.

Johnson: But it was a version that had never been maintained and updated.

Markowitz: Right.

Johnson: You mentioned Simscript II. Was that an IBM product? Had you done that under contract?

Markowitz: No. Simscript I.5 was the result of a contract between CACI and IBM. Simscript II was being developed in the meantime at Rand. So Simscript II was a Rand product.

Johnson: Oh, I see.

Markowitz: It took a good many years to develop. I think it took four or five years.

Johnson: One thing that I'm interested in is the concept that a software product is not just a set of code.

Markowitz: Right.

Johnson: One of the things I'm trying to pin down was to what degree IBM was a competitor because of the "free" software that they had. And there are a lot of people who feel that they were not a really significant competitor because just getting the code was not sufficient for most users. They really needed code that was maintained and the auxiliary services that went with a true product.

Markowitz: Right. In the simulation area IBM's offering was GPSS. I'm not sure, but I would imagine that GPSS was the most widely used simulation language and Simscript was the second most widely used. A few years later Simula came out. That may have been widely used in Europe. It wasn't as widely used in the States.

So it was GPSS first and Simscript second. I believe the reason was that GPSS was easier to learn and use immediately. Simscript took more effort to learn but was more flexible. Many users started with GPSS then switched to Simscript when they encountered GPSS's limitations.

Pricing Issues

Johnson: What was the pricing on Simscript?

Markowitz: I can't remember but it was cheap. It wasn't a big deal.

Johnson: A couple thousand maybe?

Markowitz: Yes.

Johnson: It wasn't a $50,000 package?

Markowitz: No, it was under $10,000. And you had it forever or something like that as I recall. Between Herb Karr and me, I was the cheapie guy. I believed in putting it out cheap and getting everybody to use it. But we still made money. In fact, later I got fired because of what started as an argument over how to price a new product.

Johnson: You mentioned that to me on the phone.

Markowitz: I was Chairman of the Board and Technical Director. Herb was President. We had a falling out in about the fourth year. The company had grown at a beautiful rate. We'd double and redouble our earnings every year. It was just a stockbroker's dream to look at our track record.

A lot of that came from the courses we gave. They led to a lot of the early successes. We met potential clients at the courses and we also met potential programmers at the courses. So both supply and demand were crossing. And we got some very, very good people. The company's quite large now. I understand it's about 2,000 people.

Johnson: But they're doing very little in products. They're really in what they now call professional services.

Markowitz: Yes, they're doing a lot of contract work. As we grew we got more into that. They still do Simscript II.5 courses. We gave many per year, perhaps a dozen per year. They still sell Simscript II.5. So they're still in the Simscript II.5 business. But the majority of people they have are contractors.

At the time of Herb and my falling out we had a new product coming out called Qwick Qwery – this must have been 1966 or 1967. This was a neater version of what Mark IV was doing. It was a storage and retrieval package. But it was in the days when people didn't have a lot of CRTs. You input the information on punched cards from a query form. So the question was how to price it. I wanted to sell it for $6,000. And Herb wanted to sell it for $30,000.

Johnson: And this was a competitor to Mark IV.

Markowitz: It was a competitor to Mark IV.

Johnson: Which was selling then at about $30,000, I think.

Markowitz: My recollection is that Mark IV was selling at around $30,000. And Herb and the hot shot sales guy we hired wanted to sell our product at about $30,000. I wanted to sell it at a lot less. I felt that when we got established, when we became a household word, then we could start raising the price.

This was the first time that Herb and I couldn't come to an agreement as to how to proceed. We formed a board of directors, mostly internal guys, to cast a tie-breaking vote. The problem was finally resolved on March 15th, the Ides of March 1968. Herb had 47.5% of the stock, and Jim Berkson, Vice President of Finance, had 5% of the stock. They fired me when I had 47.5% of the stock. This was shortly before we went public, so I didn't do too badly. I did okay financially.

As I mentioned before, a lot of people came to Simscript because they would start with GPSS, which seemed to make it easier for people to understand the basic concepts, and then they would find the thing was inflexible. As long as you could model in terms of particles which flowed through a network, queuing up at work stations, you could do GPSS quickly. But it wasn't sufficient when you wanted to do anything more complicated, for example, if you wanted men and machines to simultaneously be available, or if you wanted to have complex decision rules that looked downstream to see what was going on.

GPSS had some kind of a programming escape whereas Simscript had a more generic view of programming. In Simscript you view the world as consisting of entities of various kinds that are characterized by the values of their attributes. Entities could belong to sets and own sets. But you've got to program it. But that generic view was able to handle all the different kinds of problems that people had.

Johnson: Sounds like GPSS really sort of created a taste for the more complex applications and then Simscript satisfied that need as people became more sophisticated.

Markowitz: Yes, that's right.

Johnson: That's interesting. What were you providing with Simscript in addition to the code? With your own proprietary product were you providing installation? Providing a lot of training at the beginning? Was that part of the package price?

Markowitz: Yes. Training came in two forms. You could either send a person to our course or you could have your own course. So once or twice we gave their own course to some part of the Pentagon, and gave their own course to other organizations. So there was some kind of a package deal where you'd get a compiler and get your own course. And then, of course, there was always telephone support. I can't really quite call it a hot line. Even now if you have a problem with the Simscript compiler you can call Glen Johnson. There was always somebody that you could call. We were never so big that you couldn't call the guy who was in charge of the compiler.

Johnson: Was that included in the original price of the package or did you have to sign up for additional maintenance each year?

Markowitz: I can't remember. I think that at one point we may have gone from a fixed price to an annual fee. At the time I was primarily concerned with writing proposals. We started to grow and took a quantum jump when we got the contract to develop the information system for the Economic Development Administration. I switched my attention to delivering for EDA. The compiler folks were doing just fine all by themselves. Joe Annino had everything under control. On the pricing, I'm sure I voted yeah or nay, but I can't remember. The details of that just weren't big deals in my mind at that time.

Johnson: You talked about originally having the idea of needing reusable FORTRAN subroutines.

Markowitz: Yes, right.

Johnson: And then once you began to work with those you began to see that they needed to be -- would you say more generic?

Database Language Planned for Simscript II

Markowitz: Job shops come in a great variety in terms of decision rules and kinds of resources and so on. The only thing that seemed to really be reusable was these basic acts such as: create, destroy, file, remove, cause events, and so on. Now, these have come into the database world. Codasyl discovered that the world consists of records, fields and sets. But records look very much like entities, fields look like attributes and Codasyl sets are like Simscript sets with owners and members. Codasyl rediscovered the Simscript programming concepts.

I had had a theory in mind for some time that now that we had the simulation problem licked, the database problem was the next big issue. And what the world needed was an entity, attribute, and set database language.

In fact, I'll tell you how long I've had that theory. It was to be part of Simscript II. The manual for Simscript II comes in a set of levels: Level 1, Level 2, level 3, etc. Levels 1, 2, and 3 give you the basic algorithmic capabilities, like a PL/1 capability but in a different style. Level 4 tells you about entities, attributes and sets, not necessarily for simulation but for list processing in general. Level 5 added the simulation capabilities. It was planned that Level 6 would to tell you

about database entities attributes and sets. And level 7 was to put the compiler's language-writing-language at your disposal.

This was all being implemented at the Rand Corporation. My contact with the Rand Corporation got less and less as CACI got busier and busier. I tried to always be there one day a week to touch base. Rand decided to only implement through Level 5, through the simulation level, because in the logistics department of Rand, they only could see the usefulness of simulation not of the data base application.

The completion of Simscript II was overseen by Phil Kiviat who had invented GASP. I had persuaded him to write the Simscript II manual. When Simscript II was finally finished and released to SHARE, Phil Kiviat left Rand, set up his own business which developed something called Simscript II Plus. This was after CACI and I had parted ways in 1968. Phil's business failed. But they had this asset called Simscript II Plus which they sold to CACI. CACI enhanced it a bit and called it Simscript II.5. This now *the* Simscript.

Development of Qwick Qwery

Johnson: Was the Qwick Qwery product then your answer to this interest in taking the attributes, entities and sets into the database mode?

Markowitz: Actually what happened was for a long time I had thought that there ought to be an entity, attribute and set version of a database. And in particular we developed one for EDA, the Economic Development Administration. But besides this more powerful language where you could program, create, destroy, file, remove, database entities and cause real time events and cancel real time events, we decided that the people in EDA needed a query package based on the same world view. So we developed a query package for EDA and then we spiced it up and made a commercial version of it which we called a Qwick Qwery. It was the fact that within EDA they were using this Qwick Qwery part extensively, while we were still developing a more general language capability, that encouraged us to make a commercial product out of it. And, as I said, the brief history of that commercial product was Herb and I had a falling out on pricing. He won. He priced it where he wanted to price it since I wasn't around. And it didn't sell.

Johnson: So what happened?

Markowitz: In fact, CACI went public. Its stock price shot up. It stayed up for a year. Then it took a year or two before the firm started to report bad earnings and fell apart. Then later Herb replaced himself as manager by somebody who was able to turn CACI around. The company's still in existence.

The final story on this entity/attribute/set view of database management was that it was rediscovered by the Codasyl folks, but differently than I would have done it. It had the same basic concepts essentially, entities, attributes and sets. But the language was different. You had “calls” to routines (like with our initial SPS1) instead of having a language that was neatly tailored to working with entities, attributes and sets. You programmed in COBOL, and then you called a routine that would do one action for you. To do the equivalent of a single Simscript “For each of set” control phrase, it took a dozen lines of code to set up the call; come back and test to see whether you're at the bottom of this set; and if you're not at the bottom, go back and get another one, etc. Whereas in Simscript you should just say “For each order in backlog of item.”

When I first saw the Codasyl definition, I was tempted to forget about Simscript level 6. I have this other career in theoretical finance. Like here, you know, I’m a professor of finance, not a professor of computer science. So I would be off to the other career. But I still thought that lever 6 was a good idea.

Development of EAS-E at IBM

I gave a lecture at IBM’s Thomas J. Watson Research Center. The head of the Thomas J. Watson Research Center was an old friend Ralph Gomory. We were both “optimization” folks. I gave my lecture and had a tour of the place, met some people there I knew. At one point I said to Gomory, “Ralph, this place is wonderful. It reminds me of the old Rand Corporation. I'd like to work here.” And he said, “Fine. Let's put you in the Computer Science department.”

I became a research staff member and they let me do essentially what I wanted to do. But I kept saying, “Well, I want to develop this language, and I need a couple of programmers.” I wasn’t dealing with Ralph Gomory at this point. This was one layer below, the head of the computer science department which changed a couple of times.

By very good fortune an old friend, Burt Grad, spent a year at T.J. Watson on his way out of IBM. I told him, “It's just so frustrating. I've been here two or three years and nobody wants to give me the resources for Simscript SR, Simscript Storage and Retrieval.” Burt said, “The first problem is nobody in IBM is interested in supporting Simscript. You've got to give it a new name.”

So we went up to the blackboard and we discussed how do we make a name out of entities, attributes, sets and events? Well, easy. EAS-E. So that's where EAS-E got its name. Burt was helpful and I got my couple of programmers, and we finally turned out an entity, attribute and set, no events yet, language with database capabilities which works just fine. It has a small, enthusiastic following within IBM. It was published in three articles a couple of years ago including one in *ACM Communications* and one in the IBM Systems Journal.

Technically it does fine. It does everything that I'd hoped for. But it's bottled up in IBM because they've already got a couple of database languages. They've got an old one called IMS and new one called System R. The latter required a tremendous investment. So unless System R falls flat on its face they're not likely to promote EAS-E. But anyway, I got it out of my soul. So I'm back to portfolio selection.

Evolution of Reusable Code to Software Products

Johnson: Well, that's interesting. I'm intrigued by the people that first began to have the concept of code as a reusable entity, who captured an idea and put it down in a certain logical manner. And then somebody said we can use this over and over again. But then there was the process of discovering that just because it was code, that didn't mean it could be used over and over again. You said that you discovered this, too.

I've been trying to figure out which was the very first company that was formed specifically to sell a software product that wasn't also in the contract programming business. There was one that was in Los Angeles in the mid-1980s called Software Resources. It's no longer around. It was formed with the idea that they were going to broker programs that end users had spent a fortune developing code for their in-house use. They were going to take that code and resell it for you and recapture your investment. Total bust. Did not work. Because a software product has to be written in a very different manner than you write a program for specific use.

Some people seem to be able to make that intellectual jump very quickly and others couldn't. For instance, Walt Bauer told me that he had heard some of the founders of Computer Usage Corporation say in the late 1960s that there will never be such a thing as a software product. So there were people who really didn't ever think that you could generalize enough to have a product that could be sold over and over without customization.

Markowitz: At one point I would have said the languages seem to be the only reusable stuff. Maybe I'm exaggerating. But in terms of the other things that are around, we see the statistical packages. At least around universities there are a lot of statistical packages. I don't know whether corporations use those. And, of course, the spreadsheets are just such an incredible idea. With Lotus, you're sort of living inside the computer. You see all the cells.

Johnson: I think one of the things that I would like to bring out is to show that almost anybody with any degree of sophistication or involvement generally in business now has some kind of spreadsheet. We understand how that works and we understand how somebody's providing us with a set of code that allows us to manipulate information. But I'd like to try to show how that didn't just drop out of the sky.

Markowitz: Yes.

Johnson: That getting to the point of creating a product which allows you to do what you want was a long process involving a lot of people and that the intellectual idea didn't just come out of nowhere. It was built on a tradition of learning more and more about how to generalize the process.

Markowitz: I once asked somebody, “How can IMS be so bad?” And they said, “Well, within IBM, the people who built the system weren't the people who used the system.” It was so bad because they weren't users themselves. Now, with us in building Simscript, I alternated between being a consumer of simulation products, building simulations, and then building things that built simulations. I think unless the same organization, sometimes the same person, really sits and uses it, it’s difficult to get it right. Because you have the feeling with Lotus – I don't want to plug one spreadsheet over another but I happen to have worked a little with Lotus and a little with SuperCalc and a little with the old VisiCalc and the Lotus people really must have asked what do I as a user really want to have?

Johnson: Maybe that's the key. One question that keeps coming up when you talk to software people is why so many software companies end up being one-product companies. They come out with a product which initially does very, very well. Then when they try and bring a follow-along product behind it, it never does quite as well. Maybe the answer is that they financed the building of that second product from the proceeds of the first product. It's now being built by people who are hired to do this. It doesn't grow out of somebody's need. The initial products almost always come out of somebody who has a need for that application. So they go and build it out of their own need and then say, “Hey, this is something I can now sell to other people.”

Markowitz: One of the differences between the software produced by a little software company and the software produced by IBM was that IBM – at least in my impression – decided, for example, that it needed a job control language. So it would set out some broad specs for a job control language and then would delegate it. I may be exaggerating and this might not be the real history. But you get the impression that was delegated to a great mass that chunked it out.

Johnson: Yes, each doing a little piece of it.

Markowitz: That's right. It doesn't hang together. It isn't neat. Whereas a lot of the great pieces of software you feel were developed by some guy in the garage with a friend of his and they're really thinking about what it would be like from the user's point of view and get some real insight. Will you be interviewing Microsoft?

Johnson: Not at this point. I've been able to find at least two books that have been written about the microcomputer people. They've been pretty well covered including a lot of coverage in the business press. They've become really folk heroes.

Markowitz: Yes, right.

Johnson: And although I find them fascinating and I see that same quality, at least at the beginning, of addressing a need and having the ability to completely encompass it within one or two minds. So it has that holding-together quality. But what I'm trying to deal with right now is all the people that led up to that and haven't been covered. I haven't found much literature anywhere.

Markowitz: We're not the Wright Brothers – the von Neumanns and the early inventors – and we're not the latest. So we're sort of in-between.

Johnson: There's that gap in-between. But it was a very crucial gap in getting to where we are now.

Markowitz: At that time there was a company that was very good at turning out compilers. Do you remember one that had an awfully fast FORTRAN compiler? They got a contract to build a Simscript compiler, we heard. And we heard that they didn't turn it out. Simscript was too much more complex than the FORTRAN compiler than they were turning out. It was some company that was specifically in the compiler business and it had this niche of being able to build faster compilers. It would compile much, much faster than the IBM one and I suspect it did sell some to other manufacturers. Then it sold the IBM one directly to customers. But I'm just guessing. I'm trying to remember. The only part of the software business that we were paying attention to was the simulation market.

The way we expanded the company into the contract business was this: since we were the simulation experts, we expanded by doing simulation work for people. Later we made our way into the database business which was a different direction for CACI.

Johnson: They went off in their own direction.

Markowitz: That's right. Do you get *Simsnips*? They have a little house organ that they send out to thousands of people. If you were in the simulation business they would have found you out.

So we were in business starting in the early 1960s. Were there other companies around? What were the earliest software companies?

Johnson: Informatics was there in 1962. Interestingly enough, Walter Bauer had the concept from the very beginning of proprietary software. But he saw it as a tool for contract programming. In other words, develop reusable code but use it to make us more effective, more economical contract programmers. It was John Postley about two years later who had the concept of making Mark IV a proprietary product which would be sold in and of itself. So they were there.

ADR, of course, was out there in the early years. They came out with Autoflow in about 1965. But again, they were doing contract programming for hardware manufacturers and a lot of government clients. They actually developed Autoflow for one of the hardware manufacturers. I think it was RCA. And then the hardware manufacturer decided there wasn't really any market for it. So ADR converted it to run on IBM computers and had great success in that market. Marty Goetz believes that Autoflow was the first true product that was out there on the market. It was one of the very first. And, of course, it was in direct competition to a product from IBM that was "free." So he was always very sensitive to the competition from IBM. But that sensitivity varies for people depending on the type of product that they have.

Markowitz: But I assume he felt that his product was just technically superior.

Johnson: Yes. He felt it was definitely technically superior. But it was difficult to compete against a product that was inferior but free.

Markowitz: Also sometimes it's difficult to compete against a product that gets there first. You know, like the typewriter with its QWERTY keyboard. They say that you can type twice as fast on better-designed keyboards but everybody learns QWERTY. Within IBM very often good ideas will be tolerated but they won't be put forth, won't be supported, because there's so much investment in older IBM products. Like VM/CMS at first. It just was not a strategic product but it built this tremendous following within IBM. The floodgate opened and it finally came out. With the microcomputers, there's more freedom of choice. Even with IBM products you have this great selection. You look in the catalogues and you find all sorts of different editors and different products in all categories.

I think maybe we've covered what I can tell you about the beginnings of CACI and Simscript.

Johnson: I think so. I really appreciate your time.

Markowitz: My pleasure.