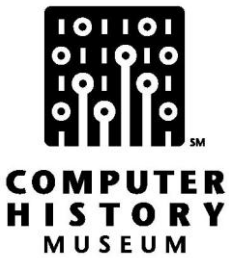

Minicomputer Software Workshop: DEC/DG Technology

Moderator:
Burton Grad

Recorded: June 3, 2008
Mountain View, California

CHM Reference number: X4697.2008

Table of Contents

Minicomputer Software Workshop: DEC/DG Technology

Conducted by Software Industry Special Interest Group

Abstract: The meeting focused on the evolution of technology from the 1970s up to the mid-1990s, and the impact that evolution had on development processes and the businesses of both minicomputer software firms and minicomputer manufacturers. In this session, the emphasis was on software companies developing products for DEC [Digital Equipment Corporation] and DG [Data General Corporation].

Participants:

Name	Affiliation
Burt Grad	Moderator, SI SIG
Ken Ross	Ross Systems
Karol Hines	Ross Systems
Luanne Johnson	Argonaut Information Systems
Paul Gustafson	Argonaut Information Systems
Joe Hensley	Argonaut Information Systems
Oscar Schachter	ACT, Creative Socio Medics
John Phillips	Creative Socio Medics
Bruce Ray	Wild Hare
Jan Phillips	DEC
Michael Mahoney	Historian
Ian Walsh	Historian
Thomas Haigh	Historian
Gerard Alberts	Historian
Doug Jerger	SI SIG
Ed LaHay	SI SIG

Burt Grad: In this session we'll be talking about the evolution of technology, what tools you used, what systems you used from the manufacturers, and your own development of technology. What did you build on your own? Did the computer companies help you do any of that? I have it broken into two periods, up to about the 1980/1981 period of time, approximately, and then what happened afterwards. We're going to look at the mid-1980s in our last session today and talk about what happened after the sale of DEC and the sale of DG? Let me start with this first side and we'll mix it up, i.e., we're not going to just do this company by company. Some of you I gather use systems products and tools from the companies you worked at. Is that the case or not? Let's start with that. Luanne, in Argonaut Systems you used COBOL. You didn't care about the operating systems or about any of those other things from the companies that you were supporting.

Argonaut Information Systems

Luanne Johnson: Each unique computer had something that had to be changed, sometimes minor changes within the COBOL, depending on the operating system.

Grad: So there were COBOL compiler issues. Paul, let me probe that with you a little bit further. Did you use any programming tools or just program directly in COBOL?

Paul Gustafson: Just directly in COBOL.

Grad: No application development tools?

Ken Ross: It's the COBOL generator that you guys want to discuss.

Gustafson: That generator was for building applications. It wasn't unique to the hardware. We generated COBOL programs

Grad: Did you build the generator yourselves?

Gustafson: Yes, we made everything ourselves. I think also our generator was done in COBOL.

Grad: Your decision to not be platform dependent meant that you couldn't really use tools from any particular manufacturer because that would make you platform dependent. Is that a fair statement?

Gustafson: Correct.

Grad: Did you build any other tools of your own?

Johnson: The filter programs. In the end we built 17 or so filter programs which basically analyzed the COBOL source code on a character by character basis and changed it as needed for each different platform.

Grad: And that program was in COBOL?

Gustafson: That was in COBOL also.

Grad: You didn't find a public product that would do either of these things for you?

Gustafson: I don't think we really looked. We kind of developed our own proprietary technique or our own subset of the lowest common denominator of COBOL and then we would filter and translate the balance of what was there.

Johnson: We made heavy use of copy statements in bringing standardized code into each program. To minimize the extra work, they actually had to be analyzed by these filter programs and I just don't think we would've ever been able to find an off-the-shelf product that did that, i.e., looked at the COBOL source code the same way we did.

Gustafson: Some of the vendors had tools to migrate you from IBM COBOL to Honeywell COBOL or whatever. Again, they were very vendor-specific and they had to be generalized to support every possible option. And we had already narrowed down or tried to have a tightly constrained set of COBOL statements or commands we'd use.

Thomas Haigh: Did targeting the lowest common denominator of COBOL impose any significant limitations on the functionality or efficiency of the software?

Johnson: Yes. Obviously, it didn't run as fast. We had to strip out any of the specific IBM COBOL compiler extensions and so on. It didn't run as fast, but with a payroll system, it's never calculation bound. It's always print bound, so it didn't matter. So we'd lose 30 seconds here and there. I even remember when they first began to make it possible to spool your print output so it didn't tie up your whole computer. That was a big issue.

Gustafson: It caused bigger issues when we moved into the relational environment, which was in the late-1980s where we wanted to support both indexed and relational retrieval. The

market was just slowly starting to embrace the world of relational databases, and some people wanted a relational database and some people still wanted to have an indexed sequential version of the application. We wanted to be able to support both and so again we used the lowest common denominator technique of having an I/O [Input/Output] module for that, which basically handled all physical I/O. One I/O module could be used for each type of data management system. In the DEC world, it could go to RMS which was their indexed sequential system, or it could go to RDB, which was Digital's relational database system.

Grad: I want to come back to the 1980s because the relational database technology was a major new factor that came into the marketplace, and it was focused almost entirely on minicomputers to start. IBM didn't come out with the DB2 until later. Oracle, Ingres, Informix and some others were running on minis. Anything else on the technology development at Argonaut that you think is significant?

Gustafson: No, I don't think so.

Joe Hensley: Reporting would be the other piece that I remember specifically. Everybody wants column 3 moved to column 1, and column 1 in column 5. When generic report writers became available, we did begin to incorporate those into the product and the big challenge there was ease of use because to try to do a demonstration of a generic report writer for the payroll department could create quite a challenge.

Grad: You say generic. These were commercially available report writers? But weren't there different ones on each of the different computers?

Ian Walsh: There was some product out of Canada that ran across multiple computers

Gustafson: Was it Zentis?

Hensley: And it evolved into something else eventually. Now they're a huge software company somewhere.

Johnson: I have something that I can't remember right now. We had the filter programs that filtered the COBOL source code but, of course, when we delivered a payroll system, we also had to deliver the job control language that would execute it. Did we have filter programs for those?

Gustafson: We had generators that would also generate the appropriate statements to basically wrap the COBOL so you could compile it all at once.

Grad: When were these mostly developed? In the 1970s or the 1980s?

Johnson: 1970s.

Gustafson: In the 1970s and they just advanced your technology. In the mid- to late-1970s, we were initially batch, and then we were character cell, and then we moved to a block mode-type full-screen, and then we moved into relational, and each one of those caused its own series of technical challenges. It became harder and harder as the technology advanced for us to continue to support all those platforms because they started to deviate more in terms of what you had to do with each platform.

Grad: You became interactive in that form. Is that what made it more difficult?

Gustafson: It made it more difficult. Again, our first entrée into interactivity was with Marketwise, which was an online system that was character based and so we again had a generic issue. Our I/O routines were generic. They were common routines and we would have a different set of screen handler routines for one platform versus another platform.

DEC Technologies

Grad: Ross Systems was focused on DEC. So my question was, from a technology standpoint, because that was where you were working, I gather, for much of that period of time, what did you use in the way of operating systems, tools, and so forth that came from DEC?

Karol Hines: When we were on the 11/70, we used RSTS. We didn't buy tools or anything from anyone else except DEC.

Grad: What were the standard ones that you used? That's what I'm trying to get. Do you remember?

Hines: Heck no. But we did build our own utilities for tracking people's time. And when we built our products, those were really tools, too, the modeling system and the database system for Impact.

Grad: You mentioned before that you were running under their operating systems.

Hines: Right, RSTS.

Grad: Did you use their COBOL to write your programs?

Hines: Yes. We used their compiler. We were using BASIC on that machine and then when we moved to the VAX, we again used their compilers, operating system utilities, and other efficiency tools.

Grad: And you operated under VMS explicitly?

Hines: VMS exclusively. Their tools worked fine for what we were doing.

Grad: That's the simplicity here. You could get all that from them and you didn't have to worry about generating your own things.

Hines: Right. The tools that we built were tools that were productivity tools for the people that we were selling to, like Model and Intact or Maps DB. We built some in-house utilities for managing the code. We built utilities for doing the installations. So, if you bought a software package, you got a module that you loaded and just answered questions and it did the installation for you, things like that.

Grad: It's interesting. Yet you built two products that many manufacturers built for themselves. You built a database product and you built a timesharing product. Why? Why didn't you just use what DEC had?

Hines: We didn't build a timesharing system. We used the RSTS operating system but we built the system that tracked the users, tracked their time and everything, so that was our own little utility. I don't know that DEC had something like that.

Grad: So you used their timesharing, it was their product.

Hines: Right. We didn't modify the operating system.

Grad: I misunderstood. I thought you had created your own.

Hines: We built what you might call tools because we used BASIC to build what was called MAPS in the beginning -- Management Aid for Planning Strategies. We used BASIC to do that and when you built a MAPS model, you could include BASIC code, plus we had a little macro language that you could use. So you might call those utilities, but they were products that we were selling. We were using them to build systems and we were selling them.

Grad: Whereas they were building things for their own use in Argonaut, you were building things that you would also sell to your customers.

Hines: When we moved on to develop software products, we did some of the stuff that Paul was talking about. We had libraries of routines, so instead of having to redo the code for doing I/O in GL (general ledger) and AR (accounts receivable), those would be standard routines.

Grad: Did you at any time in that area get some of those routines from DEC?

Hines: No, we built them all ourselves.

Grad: Why not? Did they have them?

Hines: I don't think we even asked, because that really wasn't what DEC wanted to do; DEC was not building applications. They were building systems and they were really building hardware for engineers. The fact that VMS got developed, as I understand it, was because some renegade group went off and did that. It became a great operating system for business systems. But it wasn't originally sanctioned.

Jan Phillips: At DEC we didn't do applications.

Hines: Right. They didn't do applications and didn't care about it.

Utilities Software

Grad: That's an interesting point. In the mainframe world, we had 20-some utility categories. Computer Associates eventually dominated 17, 18, 19 of those. They bought whoever their competitors were. You're saying that in the minicomputer market, DEC and DG didn't have this whole range of utilities that they provided to their customers.

Gustafson: They didn't need them. The mainframe was so convoluted and complex to use while, in the case of VMS, it was a very easy to use. It was an operating system that didn't require utilities. -- It had a job scheduler built in. There were probably 10 or 12 job scheduling systems for the mainframe but VMS just had commands built into the operating system that allowed you to submit jobs with the priorities, et cetera. So third party system software was probably close to non-existent in the Digital world compared to an IBM mainframe.

Grad: The software companies had to build some things. For the timesharing system, there was apparently no job measurement capability. You had to build your own in order to charge your clients.

Gustafson: There might have been something, but we did something that was going to work for us because we were tracking CPU's connect time, and we wanted to be able to just sort of push a button and get all that information out of the building.

Grad: Sorts and merges. Everybody needs those, don't they?

Gustafson: Sort is a VMS command.

Grad: VMS is in the 1980s, isn't it?

Hines: But RSTS had the same thing. RSTS was a very simple but very powerful operating system.

Grad: You're saying DEC and DG were building in things that IBM didn't build into its operating systems. Is that a fair statement?

Hines: It was probably the nature of the operating system, the nature of what it was. It was really more interactive than a big batch IBM system.

Grad: I know that in the utilities market, you did some work there, and we'll talk about that. But it didn't become a big market in the 1970s for the minis.

Bruce Ray: I would say no, not the way it was for the mainframes.

Grad: In the 1970s for the mainframes, there was Syncsort and ADR. A lot of companies went into that kind of space. Computer Associates went into that space.

Hines: I remember on the RSTS machine, things were so easy to do that you could really get yourself in trouble. I almost deleted a whole operating system, all of our core systems because I used a PIP command when I thought I was in a test environment and I wasn't. The command was "pip*.*/delete" or something like that. Everything was so easy, you had to be careful.

Grad: Interesting. Again, it's a very different environment you were working in. It came out of an engineering focus. The engineers could take care of themselves. They didn't expect the support. DEC and DG were expecting the user to do his own programming. The 1970s were when this started to change, where the software companies started to provide applications and packages that ran on these machines.

Data General Technologies

Ray: The Data General initial scenario was it had nothing except paper tape-oriented software, just like the PDP8, just like the PDP11 when we got our first machines. Then it evolved. You got the disk operating system so you had languages like BASIC, FORTRAN, the Interpreter, FORTRAN editor, Assembler and a rudimentary operating system. After that came the more specialized programs, the RSX, the RSTS and as you evolved to the 32-bit world, the supported products grew -- COBOL and in DG's case, there were different types of COBOL. In DG's case, there were different FORTRANs. In DG's case, there were different BASICs. In the DEC world, you had two different FORTRANs. You had DIBOL, you had the standard COBOL. So as the lines matured, you had more options from which to choose. In the languages, you often had multiple products from which to choose as well. You had the FORTRAN-plus or the standard threaded FORTRAN in the DEC world. It was the same way in DG land. You had multiple COBOLs. One and a half of them were portable across all Data General machines which caused DG marketing headaches.

Gustafson: Data General, correct me if I'm wrong, the executables were not upward compatible or downward compatible. If you created executables on high-end DG, you could not bring them down in the market.

Ray: It depended on the COBOL. If you're talking about VS COBOL, the answer is you're correct. If you're talking about I-COBOL, which is where we focused, you could take something off of a PC and run it on a Data General V machine with no changes whatsoever.

Gustafson: Whereas in the Digital world, the executable code ran on the full stack of the hardware.

Grad: Up and down the line?

Gustafson: Up and down. All you got was a faster box.

DEC and DG Database Management Systems

Grad: Were there any database management systems offered by DEC or DG during the 1970s?

Gustafson: Yes.

Jan Phillips: DEC had a database management system.

Ray: Data General had an incredibly powerful record manager called INFOS. It was not a true database management system. It didn't have all of the things that other db management systems had, but it was a very powerful super ISAM system that allowed sophisticated applications to be supported on the hardware of the early 1970s.

Grad: Was this a hierarchical system?

Ray: Yes. It was ISAM-ish with some deviations and some features that made data access very fast, very reliable and very non-portable.

Grad: One of the reasons we're dealing with the subject is that there will be a special issue of the Annals [of the History of Computing] coming out next year on database management systems focused mainly on the mainframe system. But Thomas Haigh is doing an introductory article as to where these things evolved from, the different kinds of file management systems and so forth. One of the things we haven't decided what to do about yet is there were DB database management systems on the minis during the 1970s and the question was, were they significant or not, did they make a difference, were they used extensively; I don't know the answer to that. Any of you have any information on that subject?

Gustafson: RDB had a substantial market share within the Digital world. Ultimately in the heyday of DEC, Oracle ultimately bought those rights from Digital and basically used a lot of the technology. Actually, they bought it with the condition that the engineers came with it, so that they could get a lot of the technology that actually Digital had in their relational environment on how they did two-phased commits.

Haigh: That wouldn't be the 1970s though.

Gustafson: I'm sorry. That was not in the 1970s.

Grad: That's the 1980s, so a later period of time. Somehow, the mini world evolves a few years later than the mainframe world, but in some of the technology they're ahead of the mainframe world. Timesharing is an example I think.

Gustafson: In the 1980s, on the mainframe, was IMS still in use?

Grad: Yes, although IBM delivered IMS in 1969. Actually, IBM delivered it in 1968 before it was a product. That was one case where we were on schedule. And CICS was delivered on schedule. I was part of the unbundling task force at IBM in 1969 and we announced 17 products at that time. And every one of those worked at that point in time. How about with DG? Did they have a similar thing?

Ray: I think that INFOS, judging from the reaction here, is a non-entity to the rest of the world.

Grad: But RDB was significant later on for sure.

Gustafson: I believe RDB exists today. It's still a supported product at Oracle.

Grad: What did you all use of system software and so forth from your entities?

Creative Socio Medics (CSM) Service Bureau

John Phillips: If you're talking about starting with the service bureau phase of our business, in that phase, our business philosophy was to rely as little on technology as possible. We hired programmers who had no interest in making sexy things, and built it all with as little integration and as little fancy processing as possible. Because in service bureau work, first of all, there's no time constraint really; secondly, there's no reason not to have manual intervention, if that's a very good way to control certain things; it's serial. In fact, with the lead programmer that we had, the maximum integration he allowed them to do was that the system could come up and say on the console of the mainframe: "It's time to load the tape drive." There was not that much information. The whole idea was, by doing that, you had as little technology as possible; you were totally able to utilize it and make changes and make the process different without doing software work. The whole idea here was to minimize the technical level in terms of detailed techniques and in terms of volume of software.

Grad: Were you were running under an operating system?

John Phillips: To tell you the truth, it's a long time ago, but the only languages that we could use at that time were on IBM mainframes in all of these data centers. There were FORTRAN, COBOL and machine languages. That's the only ones.

Grad: So you were running under the standard IBM operating system. You were using OS or DOS or whatever they had?

John Phillips: Standard all the way but it wasn't an issue once you're doing products. There was a kind of a meaningless aspect to it, because COBOL ran on the IBM machines and it was just never an issue. I don't remember the operating system coming up even as a discussion in any of the work we did. It was all simple COBOL programming.

Grad: You were running it as a timesharing operation?

John Phillips: No, we were a service bureau, a straight service bureau.

Grad: Good difference. Thank you.

John Phillips: That's why there was no time issue, there's no step issue, how many steps are ahead of us, and the manual intervention would be anything you want.

Grad: Did you use any application development tools other than just writing under COBOL? Did you use report writers?

John Phillips: No. All of the reports in the original service bureau work were basically sort sequences and printing, with headings. That was all done by ordinary COBOL programming.

Oscar Schachter: And there was a guy with a car who collected the reports and delivered them around the city to the various clients.

Grad: When we did an interview of Frank Lautenberg about when he was with ADP [Automatic Data Processing, Inc.] at the very beginning-- he talks about the little Volkswagens running around to deliver the payroll checks. It wasn't sneaker-ware. At least they had a car.

Schachter: Picking up the payroll data and running it. They never used Greyhound buses or airplanes.

CSM Development Processes

Grad: Let me go to the second part of this. You ran your own development shop. You built your own programs. I'd like to talk a little bit about what your development processes were, about how you went about that development work. Let me start down at the end of the table with you, John. You gave some of the rules for your programming. From a scheduling point of view, — how did you do it?

John Phillips: If you think of our philosophy and our capabilities, our technical capabilities, everything kind of moved along step by step. Until the minicomputer came along and we needed to have turnkey systems, there was no thought of productization at all. It was just what I indicated to you. Once that came along, we had to make major decisions. We had to get staff that had some inkling of what is productization because we learned over the years there are several major issues when you talk about delivering a product versus a providing a processing service of some kind. Second thing is, we made the major decision to go with MUMPS. Don't forget, MUMPS was the first, as far as I'm aware, of an open system. Everybody was doing anything they wanted to MUMPS and then sharing the information at MUMPS meetings. It was

not owned by anybody, although certain companies would try to take it over and run with it and expand it and standardize it. Picking MUMPS had a major effect on everything else we were doing. We were never going to be that close to the computer manufacturer because it had its own database, operating systems, and so on and so forth, even though a lot of people who were using MUMPS ran on DEC equipment. Over the years, through hit and miss mostly, we learned how to productize it, pull everything out of it that wasn't ours that we possibly could, interface with all the things that were becoming standard tools. That's why today, let's say, the system uses many, many commercial products in building the application. But it was all hit and miss. I don't want to give you the impression at stage 1 we had a big strategy.

Grad: When you were doing your customization work, doing the initial development, did you have project plans? Did you use project management tools?

John Phillips: The one thing that we were good at, even at the early stages, was we did have real project management; the selling activity and the legal activity and technical activity really laid out what the proposed system would be, and it was very, very well controlled. When we had problems usually, like Oscar [Schachter] and I mentioned earlier, the issue of the development work taking longer than expected, it was almost always through poor definition or touchy-feely clients that continually tried to modify the original assumptions. That was the big one. And we had to learn more and more how to either charge them for it or force it into place and make sure the charges for modifications started at a certain point and so forth. So mostly we were very good at business issues in terms of tying down the technical steps and that kind of evolved.

Schachter: John, I appreciate that but there were times when the salespeople sold a product that wasn't there. Imagine that. ...sales versus technical; there were features and functions that were great and that eventually really had to be there in order to have a complete system, but that hadn't been built yet. So sometimes a client was ahead of the curve, and sometimes a client was behind the curve. Where the client was ahead of the curve, that's where we ran into problems because we were trying to deliver something that we hadn't fully developed, fully built yet.

Grad: Did you have any formal quality assurance procedures?

John Phillips: As far as formal quality assurance as opposed to getting together on issues and problems, I would say that didn't take place until the mid-1990s if I was guessing, a long time later.

Grad: Documentation of your products, technical documentation, not user documentation?

John Phillips: Pretty good. I'd say pretty good all along.

Grad: And you had development standards that people were expected to use? That's what I hear you saying.

John Phillips: We had development standards but not as early as we should have, and it wasn't until some of the people that we brought into the company had real product minds. And to me, in an application, the product mind is so different from sexy programming, exciting programming, whatever the hell kind of programming you might talk about. The person who can look at software to be developed and say it's a product, was a major issue for us, and it didn't occur until I'd say the late-1980s.

Schachter: Or even some of it until the mid-1990s.

Grad: Wow, that's very late. Did the same people who wrote the programs maintain the programs?

John Phillips: As a general statement, in our marketplace, we were very good at that. We always stressed that we were large enough to separate maintenance from development and here are the reasons for it. That usually sold well. Most of our competitors were smaller and they tended to have a bunch of people in the room doing everything.

Grad: So you had that separation relatively early on.

John Phillips: Yes, it was all timing.

Ross Systems Development

Grad: Let's work it backwards in the room. Let's talk about development processes at Ross.

Hines: When the company started, it was just a couple of us, and we did mostly custom stuff for people. But my background had been with IBM and I got taught right in terms of how you did development, the whole development process, the whole lifecycle. Whether it was working for a client or later when we got into developing our operating system or anything else, that was the process that I followed. We didn't always have a formal project plan and sometimes our design was on the back of a napkin. Ken [Ross] and I would go off and he'd tell me his ideas on the back of a napkin. But once we got more than two people in the department who were developing programs, everything was written down. So we had a process right from the very beginning.

Grad: And you followed that? People in the mainframe world spoke about a waterfall process, a sequence from specification, design, programming, through to testing.

Hines: And that's what we used.

Grad: So you followed that. It wasn't what they now do, which is sort of build something, see if it works, change it, correct it, and continue the cycle again.

Hines: No. It was definitely a waterfall process. We used the parts of the process that were important to keeping track of everything. I was thinking about when we had formal QA (quality assurance) and I think it was when we got to MAPS DB, to INTAC, because I remember having meetings because now we had maybe five developers. They were doing both original development and maintenance, and I remember having weekly meetings where we'd go over the list of the things that we found and we prioritized them.

Grad: When is this?

Hines: Late-1970s, early-1980s. And then when we got into software products, we developed some very formal processes.

Grad: One thing for both of you, if you ran the application yourself, did that mean less need for quality assurance because it was on your own machine and you can fix it? When you send out 100 copies to people, it better be pretty clean or you have heck of a mess in terms of fixing and maintenance and a lot of unhappy people. But if you're running it on your own machine, you're making a design or program for one platform and it tends to work. Am I wrong on that?

Hines: If you're running it in a timesharing environment, you may only have one instance of it but you have a whole bunch of people running it. It better be right.

Grad: But as a service bureau, did that save you anything or not?

John Phillips: Don't forget when we were a service bureau and it was batch, not timesharing, everything was processed, examined, sent out, so those issues didn't apply in that.

Grad: I don't think it's so much of that transition, service bureau to timesharing. But there was a significant difference in that. That's a very good point. We're probably going to have a processing services meeting. If you can find any service bureau people still around alive, we're trying to pick up some of them along with the timesharing ones and that's a significant challenge.

Argonaut Systems Development

Johnson: Paul, were you doing formal design specification?

Gustafson: We used the napkin development technique. There were more features developed over lunch. I guess two things happened. One was, in a lot of cases, the development was being customer funded. So whatever we had to provide to the customer was the closest thing we'd come to a specification. If we could get away with not having to get into too much detail with the customer, we would basically not put that much detail and it would be more verbal discussions of what we were going to do, and then go build that for the customer, and then try to roll as much as possible back into the product. So in a lot of cases, we would try to find scenarios where the customer had a gap and we would want to sell them a modification because in essence they were funding enhanced capabilities of our product. There were no formal design specifications ever done on any of the products we were doing. It was much more informal.

Grad: Paul, did you flowchart them before you built them?

Gustafson: No.

Grad: You wrote code directly from your understanding of what the requirements were?

Gustafson: Right. Again, the applications we had were not that complicated. The programs were fairly modular in nature and we used a templating technique. We had standard approaches in the batch days, for how we'd do various components, and they actually had standardized paragraph names in a lot of cases, et cetera. So it was pretty easy to go in and kind of figure out where the changes needed to be made.

Grad: Is that thanks to them or thanks to you?

Gustafson: I guess it's thanks to them and it's the kind of code I inherited. The more we got into supporting all this hardware, the more we got into a kind of standardization. And we had standards at certain points in time on how to write code, to ensure that it filtered from one platform to another. It was still pretty informal because Argonaut, even after we were acquired by Ross, was relatively small in nature, such that you could still maintain tight control over what was going on and make sure the code was what you wanted. You didn't have a formalized code review but you could just take a look at someone's program and it would never get outside of the boundaries. The system always looked like single developer look and feel. If you reviewed all source code, it would look like one developer wrote it.

Grad: Is this a characteristic of doing these kinds of programs on a mini instead of on a mainframe?

Johnson: It was all done initially for the IBM S/360.

Grad: Because the applications we did at IBM for mainframes were done by multiple people teams, you had to have much more documentation. Because there were different people working on these different aspects, the pieces had to fit with each other. I'm not hearing any of that here.

Gustafson: I think it has more to do with the fact that we took a very engineering approach towards development, which was trying to figure out common design patterns and then once we came up with a pattern, take that exact same common design pattern and do it over and over again. As a matter of fact, at one point in time, I remember we were generating COBOL programs on an IBM PC that had two diskettes to merge templates in COBOL. We wrote a structure of a program with templating and lines of COBOL code were injected in certain key parts and after we got that working once then we could generate multiple print programs, et cetera.

Hensley: I remember one particular structural approach to designing a product. We decided we were going to develop a purchasing product. This was going to be cutting edge because I don't think anybody else had a purchasing product at that time. Bill Hixson told the person who was responsible for the sales order processing product to take all the S's in Sales and turn them into P's for Purchasing. Then I asked him, like turning a sock inside out? How were we going to pull the program, turn it inside out and that was obviously the challenge, turning a sales program into a purchasing program.

Johnson: You were talking earlier about this new technique, using prototyping instead of a waterfall approach. That in essence is maybe one of the first instances of prototyping because literally, there was one person who had never been a programmer, but she wanted to be a programmer, and she turned out to be a very fine programmer and stayed with these systems even past when Ross had them. The instructions he gave her were to change all the paragraphs, when there was PO to change it to SO. Then let's run it and see what goes wrong and then fix it. That was the turning the sock inside out.

Gustafson: But you laugh. There are a lot of commonalities between purchase orders and sales orders and if you look at the technique, it's a header detail type of application and you don't have to come up with how to do maintenance on line items of a purchase order differently from how to do maintenance of line items on sales orders. So we would get that technique

down on a master details type of a transaction and then go and figure out where are all the places that needed to do detail versus just master maintenance, et cetera.

Grad: I think you're differentiating the structural aspects of the program versus the application content itself, the application functionality. Until you started using database systems as built-in, you were building in all the functions on handling files, taking the reports, all these things were built. Those are common regardless of what the content is of what you did. So that was smart.

Gustafson: When you had a templating process, if you had a maintenance program header detail that worked in one area, then the only thing you were substituting was the kind of attributes that you were maintaining. So you would replicate that and use it in a different application. And there's a very, very high probability that you're going to get quality code out the other end.

Testing and Quality

Grad: Did you have any formal quality assurance procedure during the 1970s?

Gustafson: No.

Johnson: Our quality control was when the customer's checks went through, if they sent a check and it didn't bounce.

Ross: We didn't start out with 100 beta sites.

Gustafson: In the late-1970s, we would desk check and ensure that it compiled. When using mainframes, we would have needed a lot of time for testing, so we would spend a lot of time basically doing code reviews before going through the whole process to get mainframe time.

Grad: It was very expensive getting and using mainframe time.

Gustafson: Unless you had some keypunch error, there was a high probability that that code was going to execute the way you thought it was going to execute.

Hensley: Watts Humphrey is the firm believer that a lot of the best programs that were ever done were in COBOL on IBM mainframes because it was so laborious to compile programs, that developers spent huge, inordinate amounts of time desk checking versus just cranking out code and throwing it over to the compiler.

Grad: Now it's cheaper to compile it, run it, and see what goes wrong?

Hensley: What a change that is.

Grad: Bruce, during the 1970s, you were doing limited development work.

Ray: Right, it was pretty much me, so I was responsible for introducing all the errors myself.

Grad: How did you find the errors you introduced?

Ray: Usually by the decibel level. We usually had a machine available which was not an IBM machine, so we had two machines per person or, as we had more people, at least one machine per person. So we were able to do not necessarily the waterfall technique but the persistent iterative technique. And the agile programming model usually was the time between when someone saw the problem and the time you saw it. So it was a different development environment. We used the DG assembler typically, possibly one of the other DG-provided languages, so it'd be FORTRAN, or ALGOL or COBOL. DG also had sort/merge and some of the communications program but the INFOS database system wasn't very popular because it wasn't available outside of DG, but you won't see that legacy except in a few places like Lowe's or Home Depot, stuff like that. As we added a few more developers, it was still very informal.

Grad: That's interesting. All of you were pointing out that some of what they later did on PCs, you were doing your development work in a somewhat similar manner on minis, weren't you?

Ed LaHay: In our checkout programs, the quality assurance after we committed enough errors of a certain type, we finally got the clue to write a program to find that specific error. Over the years, depending on the product, we usually had a good group of regression tests to apply to any change that we did. We're not talking about design committees or getting thousands of programmers. The client wants it tomorrow.

Grad: You're in a very different timeframe, aren't you?

Ray: It's a different environment all together.

Gustafson: The quality expectations were not that high either. If you plot a chart to follow the expectations from when I first started in 1974 to the quality expectations of what a software vendor is expected to produce today, it's totally different. It has just gotten more aggressive

each year in terms of the expectation of what's acceptable now versus what is not acceptable from a quality perspective.

Ray: It's also user expectation. Someone who's just going into a computer store and going to do something with whatever it is, a spreadsheet or word processor, has a different expectation than someone having to modify inverse kinetic equations for modeling a certain nuclear plant.

John Phillips: I have to tell you, in all the years of watching this, the thing that I found was the biggest problem I had was trying to keep technical people from moving the bar faster than it had to, to execute an application. If you look at the technical development of anything -- communications, hardware, software, operating systems, databases, whatever -- and then you look at the application that you were making, they were worried about pushing the envelope on the technology faster than the application, and so it was very hard. We actually had discussions with the technical group about the programmers who didn't want to do maintenance, quality assurance, corrections, minor modifications; they all wanted to be involved in the next whatever, operating system or whatever might be exciting. That was a major business issue. It might've been very good creatively, but it was a big business issue forever. It's never changed.

Wild Hare and Data General

Grad: I'm going to move us now in the 1980s and to the mid-1990s on technologies. Take all the cases. You go through this terrible drop and then you start back up again. Were there some new tools you used, new operating systems, new environments you were working in after that?

Ray: I would say no, but they were more consistent. Just think of UNIX and the attributes of a 32-bit machine. That is, up until say 1980 -- okay, 1978 with the DEC VAX but 1980 with the DG MV -- you still had 16-bit machines being the majority of sales when the MV or the VAX was introduced. Then you had a 32-bit machine where you didn't have to compress all of your code to fit into the machine's restrictions. Now you could code and create your own restrictions. But it did open up a different and much more flexible world for us programmers. So I would say the consistency, we were using the UNIX on a wide range of platforms, Data General being the last in 1989.

Grad: It was that late.

Ray: That's when DG embraced open systems. DG had DG-UX on the MV machine but it wasn't a big seller because their own operating system was better than UNIX, as VMS was much better than UNIX. That's what marketing said and that's what the company pushed.

Grad: In the mid- to late-1980s, you were starting to build a variety of systems like your utilities type products.

Ray: We created COBOL compilers and runtimes that were Data General-compatible but ran on almost any other platform. So we were able to take advantage of the 32 bit machines and C compiler which, of course, is the best and the worst of all possible worlds, but it was consistent. That, to us, was a major shift. We could get a lot of portability just like your COBOL systems. Maximum portability in this assembler language called C and so target the broadest range of hardware. That would be the biggest two changes in the mid- to late-1980s that affected us at the new level. Again, we weren't doing anything vertically. Even our COBOL systems, what are they being used for? We don't know. There was no vertical market that we targeted; we targeted whatever COBOL was being used for.

Grad: So you would just pick it up. That makes sense. The other thing, what happened in the 1980, 1985 period as far as your technologies are concerned? Did you change what your patterns were - the use of operating systems? Did anything change in the 1980s for you?

Creative Socio Medics Technology in the 1980s to Mid-1990s

John Phillips: No, from a marketing point of view. But that leaves the technical. We had more hardware platforms, more sizes of hardware, more communications flexibilities and so forth. The biggest issue that we faced, because of all this expansion of the same stuff so to speak into all these different technologies, was productization. That became more and more of an issue as time went by. Not to minimize the switch to the 32 bit computer, the technical world had to face all those issues but they never seemed to make a major business issue in the growth of the company. It was more the productization of it so you could control the products as they went across this broad spectrum of different sizes and brands of hardware.

Grad: But you didn't change the tools you used to build your products during that period?

John Phillips: No because frankly, the basis was still MUMPS. It isn't called MUMPS anymore now. It is commercially built now and you only have to deal with one manufacturer now.

Grad: But you were able to standardize and stay with MUMPS?

Schachter: There's only one remaining manufacturer and seller of this kind of system, and it's completed probably a quarter of a billion dollars in sales last year. I tried to get them to come to this meeting but unfortunately, no one was able to come, although they were very

interested. They acquired all of their competitors so they are the last man standing so to speak in the MUMPS world.

John Phillips: So, MUMPS and productization were the principal factors, although we had to face all those other technical issues, those were the big evolutionary issues in terms of technology I believe that we dealt with.

Grad: Karol, what happens in the 1980s? How does Ross become a software product company?

Ross and Argonaut: Technology in the 1980's - 1990's

Hines: According to the computer history.org timeline that we have here, 1980 was when Ross converted to VAX and when we introduced our relational-like database. It was flat files. Then we moved into software products. It was when we got to software products that the whole issue of a very tightly controlled development methodology was really brought into play. We had formal QA but I remember like maybe in 1985 or 1986 that we actually went out and bought a suite of tools that did the whole check-in/check-out and regression testing and keeping track of all of that. It didn't prevent us from releasing something with serious errors. The client found the serious error because one of our development leads decided to put something into a release the day it went out, and he convinced the little gal who was putting the product together that it was okay. It was Digital that found the error. They were one of our big clients. I think it was a fixed assets product. I don't remember. We stayed with the VAX. I don't remember whose suite of products we bought to do the quality control or whatever, but we did go out and purchase that. We had a very formal methodology for not only development. We had an application group that did the application code and also a technical group that did all of the I/O routines. We had our own file management system. I guess we used RDB but I don't remember.

Gustafson: We used RMS first and then we went to RDB.

Hines: So we had that kind of thing going on in the library of routines so we were probably pretty sophisticated for the size that we were and what we were doing. We also at that time developed a product management group in marketing so we had the whole design piece. What the customers wanted was handled by non-technical people that were working with the customers to then feed the requirements back to the development group.

Gustafson: I think the sequence, if you start from the early 1980s, was the movement of character cell into kind of a block mode – that is, being able every single time you type a character, to have it transmitted back to the minicomputer to a block mode type basis. Our first

entrée into that was actually converting all of our systems to CICS. Then after that, we quickly again used the formalized technique and templates and we could figure out how to use the exact same code that would run your IBM CICS would also run on an HP 3000. At that point in time then, we re-licensed the software to HP, as Computer Systems. After that we relicensed it with another company and that caused us to actually get into the VAX marketplace and get an FMS version which was the name of the DEC block mode screen handler. On HP, it was View 3000 but it was called FMS on Digital. So we got a block mode transmission version of our applications, which was the same technology I think that you were using at Ross for the financial applications. That's as far as we got during the 1980s. The relational databases still had not hit the market yet.

Grad: Did you start using relational databases in the1980s?

Gustafson: Not in the 1980s. It wasn't until the 1990s.

Grad: That late, even though these other companies were all in existence and getting fairly large by then.

Gustafson: It was totally based on market demand. I think in the 1980s, we might have converted as a one-off. Our applications were under beta test for some customer that wanted us to do it. It was based upon what customers wanted at the time and we were not getting many requests.

Johnson: We had one that we converted to run the totals, too. Remember that in Kansas?

Hines: It's interesting because Ken and I both left in 1988, Ken maybe in 1989 but we realized at that point, maybe in 1987, that we really needed to move to a relational database. Ken looked at that saying "I don't think I can do this again," because it was really re-building the products. The way we had structured them, it would've been very easy to do, much simpler than maybe other products, but he just wasn't ready to make that effort.

Grad: One of the comments in my relational database management meeting was that their growth during the 1980s was primarily for query-type applications but the growth in the 1990s that just blew the thing right open was because of transaction processing. OLTP [On Line Transaction Processing] applications was what made their growth just go like that and go to the billion dollar levels where they were at a much smaller level at the 1980s. That's their statement. I don't know the accuracy of it.

Hines: Then Oracle came back to us again in the late-1980s. I don't think they wanted to buy our products but they wanted us to wrap our products around their database.

Grad: Why not?

Hines: I think at this point Ken didn't want to go through it again. When he left, he did another company. He liked his companies to get to a certain point and he'd go start another one, like he's done again.

Gustafson: The only one that was at a database level that we saw was Hewlett Packard. It wasn't a relational database but in HP, if you were running an HP application, you needed Image 3000.

Grad: I know I'm going to hear more about that on Thursday when we talk about the HP progress. Anything to add, Bruce?

Ray: We didn't use databases. MUMPS was available for Data General and it still is available for Data General. We've done some migrations just in the past couple months, so the legacy lives on.

Schachter: What happened to Data General?

Ray: Data General, depending on the press release you believe, was either merged with or was bought out by EMC Corporation.

Grad: In 1997 by EMC.

Ray: It was officially 1999, 2000.

Grad: According to the website, they say 1997. Am I wrong?

Walsh: You're wrong because I was still there in 1997.

Ray: And 2000 was the first transition year.

Grad: Then the Wikipedia entry is wrong on that one. I'd appreciate your fixing it.

Walsh: They were selling one of the storage units. They immediately stopped selling their Avian servers. The Clarion storage units they're still selling.

PCs and Client Server Architecture

Grad: Let's talk about the PCs. The PCs started to become significant in the mid-1980s. What effect did they have? Let's start with you, Bruce.

Ray: I think both DEC and DG had the denial of PC's impact on the marketplace. DG's was a little bit later than DECs but they both suffered the same fate eventually.

Grad: I know that DEC had the DEC Pro. Did DG ever put out a micro?

Ray: DG had a Micro Nova and Micro Eclipse. They were very technically good products but that didn't matter. The PCs were an afterthought. DG licensed Intel PCs. In fact, if you look at the DG PCs in the marketplace back then, or right now, you'll see they're out-of-the-box Intel machines. They use the Intel motherboards. DG begrudgingly applied some of the COBOL language technology to the PC and the other thing was word processing technology to the PC because both DG and DC had their own word processing offerings.

In the DG world, it was the ICOBOL and one other that were the main products that were migrated over. So there was a little continuity; but besides those two products, the PCs were just treated as orphan products. We didn't care because our systems ran on UNIX systems and PCs as well so it was great for us. As for DG, it was not a product line that anyone wanted to be part of.

Grad: Did it have a negative impact from a technological standpoint for you?

Ray: No, it was a very nice impact. It was not a technical impact; it was a marketing decision by Data General, which was a benefit to us.

Grad: How about Paul or Joe?

Gustafson: Just to follow up on Digital's entrée into the PC marketplace, I recall a computer called Rainbow or something like that.

Walsh: They put three PCs out simultaneously. It was a big announcement in 1981 or something. They were not compatible with each other. One was the Pro, one was the Rainbow, and one was the DEC Mate I think. It was a word processing system.

Hines: DEC Mate was totally different.

Gustafson: This could be wrong, but I thought when they killed those that Ken Olson said we're getting out of that market because it'll never go anywhere.

Jan Phillips: **"**Never, never" said he.

Hines: They weren't good machines. They didn't work very well.

Gustafson: I think he was talking about the future of the technology.

Grad: One of the interesting stories we picked up at the PC software meeting back in April of 2004 from Dan Bricklin, was that he wanted to implement the spreadsheet that became VisiCalc on the DEC Pro machine and he couldn't get one, so Dan Fylstra walks in the door one day with an early Apple II and said he wanted to do it right away. So Dan used the Apple II to do the implementation. Dan Bricklin had been at DEC working with these machines in the word processing area. It's one of those little interesting side stories that you pick up in a meeting like this and you don't pick up anyplace else. What was the PCs effect in terms of Argonaut?

Gustafson: Up to what time?

Grad: 1980s and 1990s. What happened?

Gustafson: 1990 was when Ross acquired the Argonaut payroll and HR products so that's when the bulk of the company went to work for Ross. I would say that prior to that, it really had no impact because client/server still was not a technology that was having any impact in the marketplace and we were just using them internally to do word processing mainly. PCs were being used by our customers with terminal emulators. It was a VAX terminal emulator that would emulate a VT 100 I recall and then it would emulate an HP terminal. In the 1990s, there was dramatic impact that the PC had with the advent of client/server and that kind of causes the whole change. Well, open systems and client/server kind of causes Digital's demise in my mind, but I don't know whether we're going that far.

Grad: We'll pick it up again. Had it affected your products? Did you start to move your products to PCs?

Gustafson: Had it affected our products? Well, it was client/server. It wasn't really the PC, but PCs were the enablement of client/server. I recall that I was running user groups then. They were annual user groups and over a year, it seemed to be an emerging trend and I would ask how important is client/server to you as a company? This is year after year, plotting year after year. The responses were, "not important, not important, not important, gotta have it".

Grad: It was a slow transition.

Gustafson: It's a perfect illustration that you can't believe your customers. By the time they told us they had to have it, they were already leaving because we didn't have it but if you look at the prior year, they were telling us, "No, we want this feature or functionality in the product, don't bother doing client/server."

Grad: Let me focus on that question from a technology standpoint. Did you try to implement or convert or port the Argonaut products over to work in a client/server environment?

Gustafson: Yes, we did. We had a client/server product. Unfortunately, PeopleSoft also had a client/server product and the PeopleSoft product was more compelling. It was more compelling because it was designed initially for relational database. Of course, we were a port from an index sequential file system. It had tools that maximized the use of the relational database. Actually, we saw it as an early indication in the marketplace but we didn't move fast enough as a company, at that point in time, in order to make that transition. The technique was client/server first and I can't remember if it was open systems and client/server. But first there was the relational wave, then client/server kind of piggy-backed, and then it was open systems, and they were all kind of barraging and hitting us all at one point in time. It was also where we saw the relationship with Digital change. As a joke we had commentary where up until the point of client/server Digital and Ross sales people would go to the bar together and they would basically go hand-in-hand into the sales deals, and if Ross lost, the Digital guy lost and if Digital lost, Ross lost. When client/server emerged, the Digital reps started not paying as much attention to Ross, and they would actually go in with PeopleSoft and make joint presentations with PeopleSoft and just ignore Ross. PeopleSoft supports the mainframe and all these other hardware vendors; how could you do it to us? If you go with us we only support one hardware vendor. But the reality was, they thought PeopleSoft would have a higher probability of selling, and they were willing to take the risk that the customer would choose a different hardware platform.

Grad: Let's probe this one further. Neither DEC nor DG moved into the client/server world effectively. Is that a correct statement?

Walsh: Data General had a line of servers that they launched, the Avian server around 1990 or 1991.

Ray: But that's not client/server. That's the hardware.

Grad: My point is that you were supporting their minis. Now the technology is letting you run a lot more stuff on smaller machines with a client/server structure. Did either of those companies have an effective way of moving into that new environment so you could piggyback on what they did?

Hines: One of the things that we did was work with Apple, and it was the Apple/DEC connectivity I think that was a move toward client/server. I don't know whatever happened to that. We got asked to work with them on a machine that never came out, but I think that that's what they were trying to get at.

John Phillips: Here it is [showing an advertisement]

Hines: DEC and Apple plan a VAX/Mac marriage.

Ross: Karol raises a good point, that that was part of the Ross mistake. I guess that's the best way to characterize that.

Hines: We never executed it but we spent a lot of time on it.

Gustafson: When client/server came into being, it was not the PC marketplace that Ross was going after. It was the Apple market. At that point in time, if you looked at the demographics of the typical customer who had a VAX machine, they were also very inclined to have an Apple or was it Mac by then?

Hines: It was Macintosh.

Gustafson: I guess it was a Macintosh. So we had this dilemma within Ross from a development organization: do we support the IBM PC platform only or do we support a Macintosh platform only, or do we support both? The dilemma was what tool do we select on the client in order to build the applications for client/server. If we had said an IBM PC, the platform probably would've been PowerBuilder. It was the in vogue development tool at that point in time and we probably would've used PowerBuilder and history would probably be totally different from what it is now. But instead, we picked a company called Omnis, which is based in Foster City that had a client platform that would run under both a Macintosh and a PC and it was not as effective. It was an interpretive development environment and it was laborious. It was kind of object oriented which made it difficult for developers to get over the hump to develop applications.

Grad: This was one of those major transition periods.

Schachter: Can I get a definition of client/server?

Gustafson: What is the dictionary definition? It's a portion of the application running on a desktop, which would be a PC or Macintosh and then your data residing in a relational environment on a host.

Schachter: It was different from a terminal because with a terminal, all the processing was done on the host machine?

Gustafson: There are more parts of the client/server; it's in a graphical environment. So it was the graphical basis of being able to develop a window that the user would enter data in and it wasn't mono font anymore. It now could have multiple fonts and actually colors and images on that as well.

Schachter: So it was a graphic user interface and the ability to do some of the processing on the "terminal" which would now be a desktop.

Gustafson: The sexiness of client/server is what the users just fell in love with versus them being used to 80 characters across, 24 lines down, which is a typical character cell based application.

Business Decisions to go with Client Server

Grad: Something else that happened, Paul, didn't the price points and price performance change pretty dramatically at that point in the client/server environment?

Gustafson: Price points for application software or for hardware?

Grad: The hardware cost changed dramatically.

Hines: The whole cost mechanism changed because you had a whole computer on your desk and that cost one thing.

Grad: By offloading some significant part of the functionality to the desktop to which almost everybody was going to have by that point in time, you could have a lot less machine on the server.

Ross: Less communications.

Gustafson: Personally, I'm not sure -- at least in our environment -- that these were business decisions, that these were economic decisions that were necessarily being made. If we now have a more graphical environment for our end users instead of the traditional environments, we can now deploy the application out to more users and lower our costs and do process re-engineering within these firms that will now allow us to bring down and economize our operations.

Grad: You don't think it was the IT costs, then?

Gustafson: No, I don't think it was the IT costs. It was that process engineering went hand-in-hand with the client/server implementation.

Grad: Bruce, was the drive to go to client/server because of the change in the way you wanted to set up the application, the functionality and who was going to do what work? Or was it an IT cost-driven thing?

Ray: My definition would be a little bit different. Mine would be separation of data from processing and presentation, either physically or logically. That addresses points of what has been mentioned so far but it seemed from our market -- and I'm referring to a COBOL or business market -- it could not have been IT-driven via cost savings because none of the systems that we saw reduced effective operating costs or initial expense.

Gustafson: In my mind, it was an emotional bubble that was going on. Everybody in IT wanted to have a client/server. It goes back to nobody likes to do the maintenance comment earlier. This was a sexy graphical user interface. I want to program in that. So all the internal IT departments wanted to launch client/server projects because it was exciting, it was the next wave of technology. As a matter of fact, I remember Dave Duffy was quoted saying that you would pay 10 times your license fee for implementations. Yet, that didn't keep anybody away from buying those applications.

Hensley: That was a great thud when you didn't have client/server. You just asked if you would ever be able to implement the PeopleSoft software, and they'd roll their eyes and shake their heads like probably not. But that was early on.

Ray: But from the homegrown standpoint, you're talking about the latest and greatest as always number one, more sexy; number two, it's going to get more money than upgrading a current system and that's what we saw. It's like pornography. We can't define it but we know what it is when we see it, so if you show us something and tell us that it's client/server with the GUIs and maybe the data on a network somewhere, that's client/server to the CEO.

Gustafson: This was also like the transition to the web, where the IT organizations themselves basically wanted to launch projects forgetting about the business return on investment. They want that on their resume so that they could be able to either sell themselves to the next company or whatever, but they would particularly look for projects so that they could in essence do something, either on the web or, in the 1990s, it was client/server.

Grad: Let me get more people into that. Karol?

Hines: The way the PC affected Ross, it was two things. One I talked about earlier, which was that our modeling product was superseded by spreadsheets. I walked into Intel one time to help them with their forecasting model, they all had PCs and they were all converting it to whatever, VisiCalc or Lotus 123, whatever it was. That was the point at which Ken particularly said we have to do something else because we're going to lose our timesharing base because this is what it's used for. That's when we began making the transition to becoming a software company. Then, our alliance with DEC, and DEC isn't associated with PCs, but they're building one of their own so we spent a lot of time on the Pro. I think we actually had a product on the Rainbow too, but that was short-lived. Which one of them was it that the internal module ended up being in some kind of a missile over in the first Gulf War? I think it was the Rainbow.

Hensley: They found it in SCUDs.

Hines: It was the Rainbow I think. Again, we were married to DEC and DEC says "Well, we have to have this client/server thing. It's coming." They weren't calling it that then, "But we've got to get on an interface." So now we're working with Apple, with the Lisa. We didn't ever work on the Macintosh because the Lisa was going to be the more powerful, on-the-desktop machine for the business user. That never happened.

Grad: Lisa was Apple's Edsel.

Hines: Yes. We spent a lot of time and energy working in that area and then you guys inherited it. I left.

Gustafson: You bring up a good point. At this point in time -- or I certainly know in the 1980s -- Digital and Ross did have a very tight relationship. Ross did what Digital said to do. Digital would say we're going in this direction and Ross basically just ran right with them because it was feet on the street. Digital had all these sales reps who were basically working hand-in-hand with the Ross sales reps and if this was a direction Digital was going to go in, Ross wanted to have complementary products.

Hines: Because it was working for us and it worked for us for a long time.

Grad: It worked for quite a while and then it stopped working.

Haigh: I think in the bigger picture, that's an issue with client/server. You also mentioned open systems as a buzz word. In the early 1980s, people thought spreading out the processing, that's going to be distributed computing and DEC and IBM and everyone were hyping that as the vision. But then as the idea of open systems emerges, as a buzz word, into client/server in

the end of the 1980s, early-1990s, you're buying the bits of the systems from different companies. So you're no longer necessarily going to have a DEC workstation and a DEC server with everything DEC. You use a standard PC with a Microsoft operating system, coupled with Novell Netware running an SQL model and a client application written with PowerBuilder. It's all being fragmented so your relationship with DEC is no longer a plus; it becomes a minus because it's no longer everything coming from one vendor view of the world.

Grad: But the key that Karol said earlier was with the marketing connection; Ross expected the leads to come from DEC in many cases. Is that correct?

Hines: Yes and Paul reinforced that. The point was that overnight that changed so that you couldn't just be reliant on one vendor because there's too much happening and they're not giving you all the parts you need.

Grad: Let me pick up with what happened with John and Oscar. PCs are now coming in. Client/server is coming in despite the fact Oscar doesn't know the definition of it.

Schachter: I just wanted to make sure everyone was on the same page and understanding what everyone was talking about.

Grad: That's like I put the errors in the program to make sure you were checking it out to find them. What happens here with the client/server? Does that affect you in any way?

Converting to Client Server

Schachter: I think we moved fairly seamlessly to client/server. Most of the systems that are now being sold, in fact, PC front ends, a server, a Dell server, an HP server, an IBM server, at the back end. And clients are very happily using GUIs and the whole nine yards.

Grad: From a technical standpoint, was that a major rewrite? What did you have to do?

John Phillips: First of all, I hated PCs at the time. All it meant was more modules on the PC. Secondly, yes, it was work but as Oscar said, it all evolved slowly. You had distributed processing, you had client/server, you had PC networks, you had webs. All of these in the ultimate analysis for us meant the application went to the point that now exists totally throughout the organization. But at the time, it affected finances of a hardware sale but I don't remember it causing the evolution to any major technical issue.

Grad: Did MUMPS carry over to the PC?

Schachter: Yes. We had also a client base that was largely government or not-for-profit so they were very, very slow to move. They moved much more slowly than the commercial world so we had time to retrofit the system to meet the requirements. We have one large system that's going in now. It's probably going to be the largest sale the company ever made. It's going to be a nine or ten million dollar sale to a county here in California. They're operating on hardware from a manufacturer that is out of business. There's almost no one left to support the hardware. We're going to put 2,000 users on the client/server environment. If something happens to the support people before those 2,000 users are converted to the new system, they're dead in the water. So a lot of these clients really have taken years to move forward in technology.

Grad: Bruce could simulate that machine, or emulate it. If they run into trouble; call Bruce. He has the answer.

Sizes of Companies

Grad: Before we take our break, I wanted to just quickly ask and have short answers to these. As of 1990, how big were your companies in terms of revenue? Bruce?

Ray: What year?

Grad: 1990-ballpark.

Ray: I'd say quarter of a million.

Grad: You're still a very small company.

Ray: Oh, yes.

Grad: Who would know the Argonaut number at that point?

Gustafson: The Argonaut number is the Ross number. We were acquired then so whatever Karol says-- 1990 is the year that Ross went public actually. It had its IPO.

Hines: 1991.

Grad: Did you have a breakdown of the Argonaut revenue versus the other revenues?

Hines: I don't here.

Grad: At the time that Argonaut was acquired by Ross, what was their revenue?

Hines: I don't think I have that here.

Gustafson: I don't really know. Millions.

Hines: Wasn't the sale price to like $26 million to the VC's backing Dennis Vohs? I don't remember.

Grad: Argonaut was a reasonable sized company at that point. That's what I was trying to get to. You were in the $10 million range at that point? Oscar?

Schachter: We were about $4 million.

Grad: $4 million at that point in time. One of the things I'm asking about is that the mainframe companies by 1990 had gotten to be very large. You had a number of $100 million and larger companies. By that point, Computer Associates was accumulating all kinds of companies. You had Sterling Software, you have Platinum. I don't see anything like that happening in that period of time in the minicomputer software world.

Schachter: If you're building a vertical application, or a largely vertical application, your client base is going to be a lot more restricted than if you have a system software product such as Computer Associates, spread over thousands of customers.

Grad: We had MSA which was an applications company which is over $100 million in size.

John Phillips: The big hospital software companies were hundreds of millions.

Grad: Reynolds & Reynolds was big by then. Triad was big by then. So there were some verticals that got much larger.

Johnson: It's my understanding that Dennis Vohs goal was to make Ross the Computer Associates on the DEC.

Hines: That's why they went out and started buying up all these companies.

Johnson: To answer your question, why didn't that work compared to what Computer Associates was able to do.

Hines: I think the reason it didn't work was because of their slowness at moving forward with changing new technology. What do you think, Paul?

Gustafson: It was definitely the technology play that comes to mind.

Hines: They did not think when they bought Ross that they needed to do anything with the products. But they weren't client/server, they weren't relational database, and there was a lot of investment that they needed to put in, and it took too long.

Gustafson: They didn't initially. The first couple of years, the market was humming along extremely well for Digital. Then the client/server wave kind of hit.

Grad: They were betting that it would be like the IBM mainframe market had been.

Gustafson: Another point that hasn't really been raised is that at the point in time that client/server hit, what also hit was open systems, which was UNIX. So that now meant that companies that are now running in the client/server environment really just need one big honking server to basically run their database on. So it no longer is important what the operating system is that's behind the scene that houses this database. So lower cost machines such as HP-UX or whatever, were starting to become more widely adopted by the marketplace. So that was, again, causing a lot of consternation, I believe, for Digital because that was a very fast piece of hardware, extremely good operating system, but it was like the dilemma that Sun kind of gets in, in terms of nobody really wants it because there are cheaper boxes out there and all they want is something that's going to basically serve up SQL requests.

John Phillips: I think in addition to the fact that a lot of the VARs were in vertical markets as Oscar describes. I think frankly, a lot of the VAR organizations over the years were not organizations of some other types you see nowadays, or the ones you mentioned. They didn't have master plans. They were run maybe like mom and pop shops to some extent. They were kind of entrepreneurial in their own way. And then some other things changed like in CSM. Oscar and I were just mentioning that in the next 10 years, CSM grew 10 times. The reason is, all of a sudden, as you mentioned earlier, they got into larger systems. The systems were always there. They could've worked on them. For some combination of reasons, that came later and part of it was because of the DEC hardware moves and so forth.

Grad: I was going to make one exception in the mini world. The companies that grew like crazy were the relational database management companies. They had very rapid growth.

John Phillips: But that's a broad application.

Grad: Last question and we'll have to take a break. Did the manufacturers, in this case DEC or DG, compete in the application world with any of you? They had some application programs I think, didn't they?

Haigh: I think DEC had office automation.

Grad: They had a number of them.

John Phillips: Frankly, I don't think they did originally. When they got into the "we're the greatest company" phase that we haven't gone over yet, they did select application areas they announced they were going to develop substantial systems for in conflict with the VARs that they'd been working with.

Grad: This is just prior to when they're sold.

John Phillips: That's my impression..

