Oral History of Joe Doupnik

Interviewed by:
Alex Bochannek

Recorded: July 13, 2012
Mountain View, California

CHM Reference number: X6479.2012

Alex Bochannek: Okay. Well, welcome.

Joe Doupnik: Hello.

Bochannek: Thank you for doing this oral history for us here at the Computer History Museum. Today is Friday, July 13th, 2012, and we're recording this oral history on the file transfer protocol and communications program, Kermit.

Doupnik: That's right.

Bochannek: With Joe Doupnik. I am Alex Bochannek. I am the curator at the Computer History Museum, and also present as a videographer is Eric Dennis from the Computer History Museum. I'd like to start out with you introducing yourself, giving us your full name, talking a little bit about when and where you were born, and your life up to the point of when you got involved in Kermit.

Doupnik: All right. Well, we'll finish sometime next week <*laughs*>. That is an introduction on things. My name is Joe Doupnik. I was born here in the US, but I've moved and I live in the UK [United Kingdom] now. That's my primary home. I'm an academic as Frank da Cruz is, of course. I'm a professor. I've been a professor—goodness. This is how many years? I used to be at Utah State University, used to be at the University of California, San Diego before that. I'm now at the University of Oxford. [They are] completely different kinds of institutions, to put it mildly, and similar things. I do IT [Information Technology] as a hobby. I'm a space physics person, so that's my real interest. IT is just a means to an end, which of course is how you get sucked into all kinds <*laughs*> of things. Case in point is the Kermit project—one of many—which gets brought up when we have a need and explore, and we find facilities that may or may not do the job. Then we look a little more deeply and we discovered that they're almost there. We'll fix them up a little bit, and we try to be nice people and return the favor to people, which is what happened with Kermit project many years ago. In those days—we're talking mid-1980s—when we got this started, communications were by serial ports and modems. I needed a program that could do that kind of work for me. I had an installation in California so that program came into my hands. I worked on it, put things back to Frank [da Cruz]; Frank is very clever. Next thing I knew, I was in charge of the Kermit portion of things. It was interesting because it was community involvement and people's needs coming in, and you say, "Well, you have needs, and I have needs, so why should I work for you?" You quickly discovered that it's a merging of ideas, and there the reward is in achieving something larger than what your individual requirements are, but working together to satisfy people's needs at the same time jointly. It's a peer level, peer kind of thing. I've done that. I'm currently though, just to show you how these things turn out in the end—with a small community interest company in the UK that services charities, we do IT support for charities. We're smarter; they have needs. I'm developing very sophisticated gear in the background for

them for which they get at almost no cost because they're charities and so on. Again, it's just delivery of technology, and enables people who can use it.

Bochannek: That's an interesting point. It's about the community aspect, and as an academic, giving back is sort of a natural thing to do.

Doupnik: It is. As it turns out, most people don't understand it because they're not in the environment, but this transfer of knowledge, transfer of skills, transfer of information is so ingrained in us that we can't stop *<laughs>,* which means we don't make lots of money because we don't hold back on that kind of thing. It turns out to be fundamental because people's ideas keep shifting. I've taken this into the commercial arena in a couple of ways, as well. Principally, I've worked a lot with a company in Utah called Novell. They make computer networking—one of the leaders in days gone by. I still do to this day. In fact, I'm officially a developer. The idea is to take this knowledge and experience and bring it into an environment, which then they can make money, which is fine, but that reflect out to the larger community. We're trying to do something good for the community indirectly through these channels. Open source is one thing; very unstructured people doing their own things. It's very difficult to work in that environment. Commercial is more polished and has its own *<laughs>* difficulties in working with things, but it's a means to an end, same end, and that's very much me.

Bochannek: Let's rewind to when you were in Utah. You said you grew up in the United States?

Doupnik: Yes.

Bochannek: Where did you grow up?

Doupnik: All over the country. I moved around one end to the other for years, and years, and years.

Bochannek: Would you mind giving us your birth date?

Doupnik: Yes. '38.

Bochannek: Okay. You ended up in Utah how?

Doupnik: Several of us went through graduate school together—Penn State, space physics. We decided to get a good position in the space physics community, and at that time, the University of California, San Diego was just beginning as a campus. It was a research-only institution; it was doing the same things we were. We worked there for about six or seven, eight years and it's a very competitive, harsh environment.

Don't mind the competition; the harshness is the bad thing. All the people out of the Manhattan project retired. They populate the place. We had enough and we decided to become a same-size fish in a smaller pond *<laughs>,* and we picked up and we moved temporarily to Utah as just a holding pattern. "Let's bounce." Well, these things don't always turn out the way you expect them to. We're there for a number of years and we finally split up, and I stayed because for me, it was a nice environment and it was—well my work was external anyway. Just a place to be. I stayed there for 30 some-odd years, which is about 25 more than I *<laughs>* thought to do things. I finally retired from that, and my thinking is much different, and I moved to the UK, which is much more my cup of tea, as they say. I'm continuing the same kinds of work minus the science side of things through those means, working with very talented people in a challenging environment.

Bochannek: You said there were a number of people in grad school together. You travelled to essentially three different places, and you stayed on in Utah. What was the progression of positions you had there up to the mid-80s when—

Doupnik: Oh, I came in as an Assistant Professor and worked my way up to full Professor quickly, and then I stayed at that level. I turned down administration; don't like doing that *<laughs>.* I like doing my own thing, and working with students, and working with other people. I could use my ideas rather than worrying about somebody else's ideas. That's okay to a point. You would soon burn yourself out. Then a lot of work I had done with Novell through the years in the background caught the attention of some people who were interested in developing the academic side of things. One meeting we had in Salt Lake City, oh, '85, '86—Novell chose three academic institutions, people from them, and I was one of them to work together to develop the view of academia on their products and then how that could be reflected back, which in its way is not dissimilar to the kinds of things that Kermit has done, except the products are closed. That developed, and now I'm founder of that organization. We have about 600 or 800 institutions worldwide that are members of this group called the Technology Transfer Partners, TTP, who exchange information among themselves, and with the company, and with other companies to bring it together. Again, it's to build a result based on community interest, and being articulate, and skilled, and those kinds of things. I'm involved with that, and so next week I'm spending all on that in Utah.

Bochannek: You've been involved with Novell going back to the 1980s?

Doupnik: Yes. To the mid-80s, in fact.

Bochannek: Roughly, the same time you got involved in—

Doupnik: Yes, same timeframe. It's interesting, isn't it? They're quite different technologies. One was serial port-based, the other was network-based, and one was open, the other one was closed, and these

kinds of—now to me, it's much the same and I build relationships with the key people involved so that we can exchange ideas, and I've become less focused on their products.

Bochannek: That's the next question I had was about when you discovered Kermit in '85, companies like Novell were around. There were types of networking solutions around.

Doupnik: There were.

Bochannek: What specific project were you working on when Kermit came into the picture for you?

Doupnik: I had a large science project group, part of a larger group, but I had my own group within that, and we had our own computing facility and it was all based on serial port communications of that era. About four years before that, I helped design the NASA [National Aeronautics and Space Administration] Science Internet, which again was serial port-based, but cross-country kinds of things. We were all serial port-based kinds of things. Novell as a company, at that stage, was using networks that were proprietary local area networks, very local, i.e. within a building kind of thing, and was not a good match for what I wanted to do because I needed to have people in the building, but also then reach other institutions and so forth across the country, so it was all serial port-based kind of thing. The two didn't seem to match, but the technology of networking communications was similar in many ways so that there was a great deal of crossover, which is why I was interested in Novell—trying to put things into their company and see what they were doing. Plus, many of my grad students went off and worked for the company when I was there building these relationships.

Bochannek: Were you evaluating different options for this serial line networking?

Doupnik: Yes. One looks at the available possibilities and what seems feasible, but it's not quite there yet. Maybe develop—it's exploration. I've been involved with some very complicated things recently, and I've had to explain to people how I got where I got. It's interesting because with technology there are lots of unknowns, and so we start off exploring what looks like a possibility and it turns out to be not very good. We look around, and we explore some more, and we had to really keep digging, and none of them is going to do the job and so we have to stand back, say, "What are we going to do next?" The analogy I use with people is that we were building a railway from the middle of the US to the west coast, and back in the old days you didn't know what was between the two. You'd say, "We're going to build a railway. That way's north, and west is that way. What are we going to do?" The answer is, "Well, we're going to send out some scouts and figure out what the length of land is, send them out for several days. I, build in that direction, and then we get there and we do the same thing over, and over, and over again." Pretty soon, you end up at the Grand Canyon by mistake *<laughs>*, oops, and you had to back away and start over and think. The analogy I give people in this kind of thing is explore a tree of possibilities to some depth and then prune it to the realistic, but keep a garden because you may need these little bits of

technology that come into play later on. The trick is to keep moving and melding, and until you finally get to the objective that you want. It's a very scientific, deductive—it's not a question of intelligence or anything like that. It's a process of thinking your way through, exploring, being thorough so do you do take things to the bitter end, find out what's true versus assumptions, and the put things together, and then try to build on top of that and make progress. It's very much a way a lot of the open source projects are, except they don't take things to the depth they need. When ordinary people get them, we discover they're only a piece of the puzzle *<laughs>,* and we have to then combine this with that and figure out what they can and can't do, and make them do the things. It's an integration—it's a knitting process.

Bochannek: Do you recall the options you evaluated before you settled on Kermit for that particular project in, I believe it was '85…

Doupnik: I don't. There were, in that era, a number of programs that ran on CP/M [Control Program/Monitor]. There were some Telnet communications programs that were there. I can't recall the details, but of them, Kermit was obviously a member and had the most flexibility. It was doing two things I needed: one was transferring files, and the other was allowing terminal-like communications. Both were essential. It seemed to be able to do the file transfer portion a lot better than the other programs. I didn't know how much better, but it was certainly at the stage I was thinking things through, considerably better. That was the choice to explore. As it turned out to develop, to make it be more like what I needed. That's true with other thing as well; we keep doing this—I do anyway—these kinds of development probes and a lot of them get cut off and thrown away. Kermit turned out to be synergistic in the sense there were enough people behind it, interested, and so on to keep making it better.

Bochannek: What was the environment for which you needed Kermit? Was it MS-DOS [Microsoft Disk Operating System] on—?

Doupnik: Yes. In those days we had MS-DOS-based machines in one form or another. I had a large mini-computer, a DEC [Digital Equipment Corporation] PDP-1170 that was servicing the larger group, and we all had terminals, physical terminals that communicated to it. We're doing all our work on DEC's operating system at that stage, but I needed to be able to move things from the more local machinery we had—very primitive in those days. Very quickly the PCAT [?] came along, and we finally had real computers. Then that built up based on those kinds of computers and that kind of working environment. It got to the point where somewhere along in this line, and somewhere in the '80s the word processing capabilities were needed by the group as a whole—my secretary among others. We didn't have anything, so I looked and there was an open source program called Runoff *<laughs>.* Really has quite a long history in that kind of thing, which was most of the way there but not quite. I again got ahead, rewrote a lot of Runoff, and then distributed it around the campus as a matter of fact, saying, "Well, here's a way that you can have individuals type things and it goes to the computer, and then you print it out, and that's the end of that, versus typing it all over by hand." Everything was serial-based at that time. It wasn't until quite

a lot of time later that we got the networking portion—things like Novell had done and others—to come into play, and then finally Ethernet came in, and we all jumped up and down.

Bochannek: The PCs [personal computers] would be hardwired into the 1170?

Doupnik: Yes. That's right. We had a building, and a lot of rooms, offices and we ran serial port cables all down to the central computer facility. It's a lot of wire.

Bochannek: Do you recall the operating system that ran on the 1170?

Doupnik: The answer is no. It is the progenitor of DEC's DMS [document management system] operating system, but for 16 bits. I can't remember the name; I spent enough time inside of the dumb thing. I worked with DEC also, developing that on the inside—beta testing, and all these kinds of things just to get it to where I wanted to go.

Bochannek: You discover Kermit. Was it already around at the university? Had you heard about it from other people?

Doupnik: No. I don't know quite how I stumbled across it. In those days, things like bulletin boards and so on were the popular way of exchanging information, but I did, and I downloaded it—probably Simtel20 or some other archive of that kind, like a lot of things, and explored it and obviously I was beginning to do what I wanted to do. A problem in those days was the software was very poorly developed; it was not the kind of software we call production material today. It was more for specialists. You could see everything that's there, and if you didn't like it you could change it, but it wasn't polished as a deliverable where you just click and say, "Okay." Life was much different in those times, and it was certainly true with Kermit.

Bochannek: At what point did you contact Frank da Cruz and the folks at Columbia?

Doupnik: I think I worked on it for some months on my own as I do. I made changes, improvements, and then simply said, "Look, this is all very nice. I should give it back." I contacted Frank and said, "I've been doing these kinds of things and here's a copy of it. Hope you have a good time. Bye." Some people respond to that in different ways, and Frank said, "Well, that's really great. Blah, blah, blah, blah. We don't have anybody who can handle it right now." <*laughs*> Right. Here's a rabbit hole. Would you like to crawl in? <*laughs*>

Bochannek: This was in 1985 I believe.

Doupnik: Yes. Somewhere in that vicinity. That's correct.

Bochannek: Did you interact with Bill Catchings? I think he might have left already at that point—

Doupnik: I think he left. Most of the communications was by two forums, just to give you an idea of how people interacted. One was some direct email. In those days, we didn't have the internet that we have today and so on, but we had <*inaudible*> to do that. The other one were—today we call forums, which is a public exchange. Usenet was the thing where messages could be posted, and then sent out as news items to all the subscribers, and then people contribute and it comes back, so it takes half a day to get the turnaround. I used both, and Bill was involved in that as quite a few of the other people, Kermit people through time—to do that. By the time I came into it, I think the Kermit project had largely fallen squarely onto Frank's shoulders to carry on, and then of course he had many pots to stir because of the different operating systems.

Bochannek: The three names that I have here in the history of what led up to your work on MS-DOS Kermit were Bill Catchings, who did the conversion from the CP/M Kermit, and then Daphne Tzoar, and finally Jeff Damons.

Doupnik: Yes. Now those latter two came in much later in the game, into things. I caught it after it had been converted from CP/M into MS-DOS style, and picked up from there. I remained with that for years, and years, and years with things. Finally the organization grew a bit and Daphne came in to bring a little bit of ordinary English <*laughs*> to things, and they began to write up stuff which we badly needed. Curiously, as we go through these things, we could talk technical to other people, and Frank and I both talked in meetings, but the thought of making a simple understanding for a much broader audience really hadn't entered our minds. Again, the specialist territory; bit of blinders on. Daphne came in and said, "We could write books on this." [I said] "Books? My goodness." <*laughs*> I'd read the textbooks; I know what goes on with that. Today, we think of things much differently, which is the public presentation almost comes first <laughs>, and then you get around to actually doing the work. Again, the roles change from the pure technical, we're having a good time and, "Oh, you'd like a copy?" "Yes." "Here's three." To this business of, "We want your attention, and we want your," whatever it happens to be brought in, and then develop a product and make it go. The fact that we're a product and an open source too is a sort of oxymoron, but there we are.

Bochannek: You mentioned several times now the difference between the specialists and the tools for the specialist, and the published product. Sorry; the polished product and published specifications and so forth. What [I'm] curious about is if you—when you look at Kermit at the code level, it is a very interesting piece of work. The different versions of Kermit, of course, being vastly different see Kermit being very different from some of the other ones. Can you talk a little bit about at the time the development environment, shall we say? I mean MS-DOS Kermit is an assembly language program; how did you go about generating different versions, testing it, distributing it? Just what—

Doupnik: I had no idea what I was getting into. Now, let's focus on the language. I speak a lot of computer languages just to—you know, you need it. You figure it out and go from there. That was fine for one narrow channel, and then two things happened. One was the multiplicity of versions, the variants—not just for the IBM [International Business Machines, Inc.] PC, but for various other kinds of computers came into play was one, and the second was mission creep where these suggestions are, "We could do this, or we could do that," came in and so on. Neither was designed into the system from the beginning, so you don't have this overall view and then you start developing from there. The variants were adaptations of the standard IBM piece, but with different communications hardware and other nuances. It's very difficult to test that, so one uses a crystal ball and then hopefully finds someone who has the real machine that can test it, and then bring back that information and spread it out. I had a view that we would have these levels, release levels, and then develop the main piece and then try to get the surrounding complementary pieces developed so we can come out more or less at the same time and things. It means working with people who had these other machines. Mission creep was perhaps the most interesting because that meant creating software, and in very controlled environments. We had small memory space, very few facilities for our high-level languages and so on, and already MS-DOS Kermit was an assembly language, so you started to stay there. As we added features like a command land facility that people could program themselves, they liked the word "Foobar," so they want "Foobar" to mean something else. You say, "Fine. You know, Foobar is this." You can define an instruction, create it on the fly. You can even use nicknames and type "F-O-O" that figures out you mean Foobar and these kinds of things. You had to create the computer science notions and implement them in code. Implementing in code is pretty straightforward; to me it's almost one-to-one. But in the controlled environment, a small memory space, very few facilities from the operating system, it became a challenge, *<laughs>* to put it mildly. One finds ways to do it. Then the communications portion of that mission creep came into play. Now, there's a bit of story here. We had been in serial port, various kinds. We had a protocol that ran over the top of that. Times change; the internet arrived, which may be on your list of questions, and so we needed TCP/IP [Transmission Control Protocol/Internet Protocol]. We needed—well, we didn't need, but AT&T [American Telegraph and Telephone] thought they needed NetBIOS [Network Basic Input/Output System] support, what today we call Windows Networking. These pieces had to come in and be built in and made to run as part of the invisible infrastructure to make things go, which again is a good computer science problem.

Bochannek: How much time have you spent on this? You still had your professorship in the physics department.

Doupnik: Oh, yes. I still had a full-time, regular job as professor, for which they pay me. I'm the science person, so the interaction level is very high and so on, and we had these other little side things *<laughs>* going on. You spend most of your time doing all these things as a combination. Some people are like that, have always been that way. I am that way to this day *<laughs>*. I am up to here with things. I wouldn't be happy without it. And *<inaudible>* divide your time and then try to make progress, and you sort of switch between them and so on. It's a long-term continuing effort, not just a one-off.

Bochannek: Where did the TCP/IP stack come from <*inaudible*>

Doupnik: Came from a fellow at the University of Waterloo. A TCP stack sounds very complicated at first. It can be; lots of pieces and so on. I understood TCP rather well, so I knew what I needed, but I didn't want to have to write it myself, particularly in assembly language. I'm trying to remember the name of the person who had it [Erick Engelke]. A really good guy, very smart guy. We asked him if we could have his TCP stack. It was written in C, and he said, "Yes. Fine. Great." It became a task of now integrating this thing, so we had a C in assembly language, no real big deal for me. Then correct the stack to do things the way that we needed them to do it, and fit it all together in one tiny compartment. TCP stack is—I think I mentioned it one time. It's about 17 kilobytes <*laughs>,* the whole stack. It's amazing what you can do that way. Through the years I've taken the Kermit program in classes, networking classes, and said, "Right, this is the program. It's got lots of pieces. You're going to gag on things, but this C portion is the stack." We're talking about networking in TCP/IP, so why don't we use this program as a test tool? Go in and find out where the piece of code is, and make changes to do something interesting. You now begin to see the consequences of your changes. You can see all the pieces working together, and that worked well for many years.

Bochannek: What about the other networking protocols that ended up being supported like NetBIOS you mentioned already, or—

Doupnik: Yes. NetBIOS came in. AT&T needed to have a presence of networking on the desktop. They had UNIX machines. NetBIOS came along largely from them—a community of companies—and so they asked me to write the piece that would work on the desktop, which I did. It was an interesting challenge to do. Of course, that's all old-time stuff now, but it's evolved to CIFS [Common Internet File System], and Windows Networking. That was a completely different environment, whereas the program that I was working with was really just a frontend for the communications stuff, and the focus was all on the communications to make it go. The thing we couldn't do well—we can today but couldn't then—was to integrate the communications program with other applications such as a word processor or whatever it happens to be. The system wasn't built to multitask that way, so we just had a communications program. Files moved from A to B, or we talked—because we had terminal communication, but we didn't have a working environment the way we did with Windows or Macs. That had proven to be quite an interesting other challenge when you have to integrate these other major programs. You figure out those interfaces, but we didn't.

Bochannek: Novell networking, I suppose, was also supported? Or was that through the NetBIOS interface?

Doupnik: Novell's networking at that time was very proprietary. Like all the other companies, there were a lot of variants out there, each one of which was proprietary. You couldn't touch. They didn't want you to see the inside. I couldn't see the inside, but I knew how the things worked. My crystal ball was in pretty

good shape then, and so I could talk sense to them about what technology should do versus what it is doing. I helped bring them in to the TCP/IP world kicking and screaming <*laughs*> in those days because they had their own way and so on, and then they finally made the grade on things. I couldn't bring Novell's work into the Kermit work. The two were quite disparate to the way they were handling things, which is too bad because some of the networking, early networking by Novell and others was all very primitive. You send and hope it gets there; nobody checks, and in today's world we know it doesn't always get there. Kermit had the capability of saying, "Oh, well we sent something. It probably is not going to get there. What are we going to do about it?" in the code, and make things go smoothly. A design of Kermit was predicated on things will be bad. We must have the system to take care of all this badness so the user doesn't see all that stuff. Whereas the other guys were saying, "Oh well, we just hope it works." <*laughs*>

Bochannek: In a way, we see this merging of two different generations of data communication and networking, two different worlds. You have the serial line, point-to-point. You don't know if the bits even make it; the view of the world, which is what Kermit was designed for. On the other hand, you have high-speed LANs [local area network] with packet-switching and network protocol stacks. Why run Kermit over those networks? Why not use FTP [File Transfer Protocol] over TCP/IP.

Doupnik: Yes. That's a good question on those things, and the answer is FTP is very limited. You can't switch directories and see things and so on, and yet it's present. Kermit provided terminal emulation—you know, MS-DOS Kermit, and provided file transfer that was robust. Those are those two things. It could scroll through subdirectories; say I want to transfer a whole group, and it took care of that. Whereas FTP was a little more clunky; still is to this day. It hasn't budged in the slightest on things and had, of course, terminal emulation wasn't there. Telnet was present, but we needed a terminal emulator to express on the screen what was going on. For a long while, we ran Kermit as a user frontend, the user facility underneath the covers. We thought we were talking to the local disk, and of course now we're talking to the network kind of things, and those two worked together quite well for a very long time. I find I retired the MS-DOS Kermit mostly because it's so awkward to try to run today under Windows DOS emulation. It's all largely—they've changed what's going on, but I've used it until maybe three, four years ago.

Bochannek: We had a really good point earlier when you talked about why you chose Kermit, and we talked about it right now again, which is the sort of duality of terminal emulation and file transfer, and often when you describe Kermit it's hard to put your finger on what it actually is. It's a protocol for file transfer. It is a communications tool. It's got its scripting language. It's got the terminal emulation that's a bit of a Swiss Army knife tool.

Doupnik: It is very much that way. It's a mission creep of the worst kind <*laughs*> if you like because you say we have something that works, which is a file transfer that works reliably, and then we can stick piece on, that piece on, and so on and so forth, which is all very true. You're creating an environment in which to get work done.

Bochannek: Let's talk about the terminal emulation because I'm curious about that. I understand that the CP/M version already had some terminal emulation—

Doupnik: Yes, it did.

Bochannek: —support.

Doupnik: Yes, it did. It had what we call VT102 DEC terminal. VT102 class emulation. It turned out that terminal emulation is a very complicated subject <*laughs*> to put it mildly, and what was there was insufficient to do a good job under the environment which we saw things, so I had to learn a good deal about the subject. Terminals are state machines. You send a little burst of information, which the other machine understands and deals with it, and then finally you see a display. I had to learn a great deal about what kind of terminals were there, how they worked in terms of their definitions, and then how I could implement those kinds of things—and then tests to make sure that we reproduced what a physical terminal would do under the same conditions. The target was not a VT102; it was more along the lines of the VT220, and then finally a 320 that's sort of the end of the line on that kind of thing. I had 320 real ones, and make it do fancy displays, and even some elementary graphics for IBM PCs, which the real terminals couldn't do. And to do it well. It took a lot of understanding of how the real terminals work. You look at the specs and you have to try to get to the real terminal to see how things work, and then figure out how to handle things. While we're doing this, Frank again—mission creep—says, "Oh, well, now if you've done English, why can't we do some French? German? A bit of Spanish. Oh, we've got Hebrew and what the heck?" We got into the subject of representing text in different languages—code pages, these kinds of things, which Frank has quite an interest in. He's been quite knowledgeable about that stuff. I wasn't, so I had to learn my way and the hard way, slowly creeping forward. Once we understood that we had a translation, this bite stands for this graphic here, and that graphic there, and a way of saying which one to choose across the wire, and then we decided we could build that into terminal emulation and make things even fancier than the commercial products. You mix English with Hebrew so the direction to write changes, and the code—the glyphs change. All that could be done. If it could be done, I could do it.

Bochannek: You just made some really interesting comments here. You said that the terminal emulation was really something that you needed in your department—

Doupnik: Yes. That's right.

Bochannek: —presumably so you wouldn't have to have a separate program or a separate device.

Doupnik: Or a separate terminal, in fact. Yes. Right.

Bochannek: Convenience, cost savings—

Doupnik: It's more convenience than cost savings. Money was not the objective. We're always starving, but we had money. We had the ability to do everything in one place.

Bochannek: Okay. Now you primarily were interested in emulating the terminals that you're operating system on PDP-1170 supported, but over time MS-DOS Kermit also acquired additional terminal emulations, presumably of devices that you didn't really care about that much.

Doupnik: That's true, but I'm being a nice person *<laughs>.* You're in the system, and if you're the responsible person—

Bochannek: Were they contributed from other users?

Doupnik: Not really. The information would come in and I would write the code.

Bochannek: Okay. Then somebody else would test it and—

Doupnik: Yes. That's right.

Bochannek: Okay.

Doupnik: Which works well because you have expertise in one area, and then you have expertise in another, and you work together on these things and that's fine.

Bochannek: However, the international character set support was not something you needed.

Doupnik: No. It was something that Frank wanted.

Bochannek: It seemed logical since you had worked on terminal emulation to include it at that level?

Doupnik: That's right. He was really quite interested and hot on the subject. We spent a lot of time discussing what kind of code pages and ways of representing things and so on. I more or less followed along doing my share of the pie of the things to make all this work. It was interesting because even to this day, people are unaware of how the code underneath represents these different glyphs and so on. I keep

explaining it to people. It was not that they didn't have a direct interest in it; it was curious. Practical interest. It's like, "Oh, wow. Yes. We can do that. We can make things turn around." Then it was sort of hobby-level interest in these things, but it didn't have a direct need.

Bochannek: Did that result in any communications with Kermit users who had these needs with you directly? Or did this usually go through Frank?

Doupnik: Worked both ways. A large portion went through Frank, but the Usenet forms was a common way for many people to express their opinions, and we saw it because they didn't know who we were and they just put it out in this public forum. Some people knew the email addresses of people like myself and would send messages to us, but it was primarily Usenet and what Frank saw from his collection of interactions that determined what the goals were as we went along.

Bochannek: Now, there are many levels to character set support, right? I mean there's—

Doupnik: Yes. That's right.

Bochannek: —the display issue on the terminal emulation, there are things like file name issues, there are many of these.

Doupnik: Yes, there are. It's a complicated subject, no question about it. Not only do we have different characters, but we have multi-byte characters these days and so on in life that becomes very interesting indeed. We have those problems to this day in quite a variety of things, and the like. They have to be codified, and people agreed to the system so that we have some way of representing those little nuggets of information so they can be reproduced on the other side, and it's very difficult to get people to do that kind of thing. The ISO [International Organization for Standardization] committee came and did a lot of work on how to represent glyphs in different languages, and the code that corresponds to them, and how to handle some of this stuff, and that was the basis for a large portion of what we did. I still have a book about this thick of the ISO different languages, and the glyphs that go along with it.

Bochannek: Another aspect of MS-DOS Kermit was the scripting language support, and related to that, I'm curious how certain features in MS-DOS Kermit made it into versions of Kermit, specifically the interaction with C-Kermit.

Doupnik: Yes. The scripting language was strange. It came to me mostly there from the work that other people had done, and then I had to modernize it internally in the code because the patterns had to be more or less—and they could, the way I thought, rather than the way they thought. Then Frank came and kept coming along with suggestions on things. I've thought about this recently. If we had been able to use

hindsight like we could do today, I would have designed a much more comprehensive system, the computer science comprehensive kind of thing, but we were working from the other end. <*laughs*> "What do we face today? Oh, my goodness. Oh, jeez." We hadn't the forethought to design a complete system and then fill it in as we went forward. Life was not that simple. I had to invent as I went, techniques, and then they got reused and amplified a bit—command substitution and all those kinds of things came along naturally, but it took a while to figure out how to do that kind of stuff and it was more—it was 80 percent computer science, 20 percent code, this kind of thing. Again, because my horizon was limited, I was just trying to accomplish the nearby goals that I hadn't the time or energy to think it through more completely and I would have built a regular parser and put in a formal parser, a grammatical lex analyzer, but I didn't. I didn't want to get dragged into that, so I had to do everything by hand. It's amazing we got as far as we did <*laughs*> quite honestly.

Bochannek: Was there any other system for which you took inspiration? I mean are you familiar with the DEC operating systems..

Doupnik: A lot of DEC operating systems had similar features of command. Many—command substitution and partial things, and then macros. There's nothing new in what we were doing. It'd just bring it together in a particular focus. I don't think we invented one thing. The Kermit protocol is about the only one thing that got invented out of this whole business. That's its own.

Bochannek: You worked on improving that as well. I believe it was the sliding window support?

Doupnik: Sliding windows. Yes. I teach computer networking and so on, and the sliding windows the way of making things work smoothly with loss, and how you cover it up. We sort of did it—I want to say ad hoc, but by thinking in short-term hoc we handled this kind of problem, and it turned out that we were doing a better job of it than the TCP community to this day, as a matter of fact, and the like. That was a nice piece of work, largely forgotten now because the material has not been written up in simple English for people. My students have gone through it, but that's about all and these things, which is too bad because then there's a bit of knowledge that maybe—lots that had to be reinvented. It can be reinvented; it's not that big a deal. It's too bad we can't get these things in distilled, is my view, to the things. None of us have time to do that. In fact, going back to the old code is probably very painful <*laughs*>. Why we would do this stupid thing, and so on, and so forth. Takes a while to digest it, to distill it, to get the essence of a dozen, maybe, key ideas that come out and illustrate them, and explain why they are interesting in their own right. I don't think we have the time to do that these days.

Bochannek: How did you think of yourself in the context of the Kermit user and/or developer community?

Doupnik: Just one more person.

Bochannek: It seems to be MS-DOS Kermit was one of the more successful versions of Kermit.

Doupnik: Yes, it was. Partly—well, not because it was so great in its own right, but because it was becoming a PC world, and so that was the natural focus of things. I don't think I'm anything special at all, and very much peer kind of thing, a level. Everybody's contributing, maybe just complaints, but they're contributing in their way. Prestige was of no interest at all to me, or any of these personal kinds of things. Getting something done that was useful and other people thought of as useful, and we thought we were making progress on things.

Bochannek: Distribution of MS-DOS went through Frank in Columbia, correct?

Doupnik: That's right.

Bochannek: Okay. You did not distribute it directly?

Doupnik: Correct. We tried to make sure that that was the case. Frank and Columbia are the originators, maintainers, all these things. It's right and proper that they act as a focus for this kind of thing. If you get into the self-distribution, if I decide that I wanted to distribute it, then we have a fork in the project, and we have all this competition, which is silly. There's no point in me getting involved with that kind of thing. Again, if we work together, we work together, not separately.

Bochannek: How did the MS-DOS Kermit features make it into other versions of Kermit since there was no code reused between MS-DOS Kermit and C-Kermit, for example?

Doupnik: That's right. Mostly through comments, I think, or discussions in the group—how we handle certain issues and so on. There was a lot of discussion back and forth that gave people the clue that they needed to do this, I don't know what this happens to be, choose an example, and then see if they could implement it in their own frame of reference and then keep the command syntax consistent. Frank was using C-Kermit as sort of a benchmark standard for doing that kind of thing.

Bochannek: Was there any forking of MS-DOS Kermit at all while you worked on it?

Doupnik: There wasn't. Nobody really tried, and I think because it was an assembly language, it didn't have a manual that explained all the little bits and pieces in great detail, so we really had to work at it to understand how it worked, and no one had that kind of level of interest, which is too bad in its way because whatever we do, my opinion, we really want to put it in the hands of others. That's the only thing that counts in the end. We can have a good time, but unless they have a good time, in the end we go on.

It's the end, and that doesn't make good sense. Somehow it's transferred, and complicated things are difficult to get across. There's no question about it. Today, literally, I'm involved with building complicated server systems and I've had to go to great lengths to document it in ways that people could begin to understand how the pieces work together. If they did wish to get in, they have some navigation information. There's no end to how much you can document and that kind of thing, but I felt it was necessity in my slides that—I call it "hit by a bus syndrome." <*laughs*> What happens if I'm not here? Who's going to fix this thing? It's more than that because it's not just fixed; it's actually getting information, useful information into the hands of others who can use it.

Bochannek: You worked on MS-DOS Kermit in one way or another between about '85 and '95?

Doupnik: Yes. Something like that.

Bochannek: At one point, Jeffrey Altman started working on the OS/2 version.

Doupnik: That's right.

Bochannek: Did you have a lot of interactions with him at all?

Doupnik: He and I talked quite a bit, but he was pretty much running his own show. Frank and I talked quite a bit about trying to do an MS-Windows version of all this stuff, and that would turn out to be a huge amount of work, and our backgrounds were not suitable. Jeff was much closer to that.

Bochannek: The deal was to Kermit—eventually turned into K95.

Doupnik: Yes. He had the frame of reference and he began to understand the toolsets and all these kind of things, so he was going to handle that particular regime, which I'm very grateful for <*laughs*>, thank you very much. I didn't want to have to master that myself. He came in and started doing that material, and then OS/2, and then the Windows version. The Windows version worked quite well. There's a lot of synergism in terms of what it looks like at the command level, but the inside's completely different.

Bochannek: The feature set in terms of terminal emulation and initial character support, scripting language support, the major things that MS-DOS Kermit really brought into the Kermit world—they're all in the OS/2 and then the K95 version as well, but again, no code reuse?

Doupnik: No code reuse. Yes. That's right. The ideas transferred, and people—you get <*inaudible*> experiment of one, see if the other side behaves the same. That's correct. A lot of things work this way.

The origins of a lot of these things are well outside of Kermit project <*laughs*>. Command, we're not even aware and we begin to implement them and make them happen in our environment and that's the way it should be.

Bochannek: Can you trace the trajectory of your involvement with Kermit for me from about the mid-80s to the mid-90s where there's specific milestones that you will point out?

Doupnik: Yes. Good question. The answer is; I don't know now. I knew that in the very beginning that Kermit 2.2—I want to say 6, 7, and 8, that series—were difficult because I was learning a great deal at the time, and I was very, very busy. Those matured and then Frank decided, well, we're going to call it Kermit version 3, MS-DOS version 3. At that point, it started looking like a product to me. It got recognition, we had to have polished, and complete this. Then as new features came in principally from Frank, the need, then we marched through the minor versions 11, 12, 13, and 14, and so on, and they became very much, in my eyes, a product. It had to look right, work right, had to be fixable, automatic in various ways, and that became quite a chore. A lot of the creativity, the new ideas stopped. We added features. We became really quite a large amount of work to do. My time really just—I was completely saturated at that stage. I became much more involved with externalizing Novell-like networking-like kinds of things, and so the interest in the Kermit project fit, and of course then the technology spelled the end. Windows came along and took care of that with things.

Bochannek: How did the MS-DOS Kermit book come about?

Doupnik: Christine did that.

Bochannek: Christine Gianone?

Doupnik: Mm-hm. I didn't. That's largely her work and Frank's. I did some editorial kinds of things, creaked and groaned <*laughs*> saying I wish they hadn't said that, but that was external to me. I could have done it, but that's not the way things were being played out, which is fine. She was good at that kind of thing, so why not?

Bochannek: Who else worked on MS-DOS Kermit on the coding itself? It sounds like you took in suggestions, bug fixes, patches.

Doupnik: I was about the only person, at least—there's a lot of comments from other people, and I assimilated them. I had the moral responsibility to incorporate and make sure that things got in there that should, and rejected, but I was sort of the shepherd for that, and I had no problems with other people doing things. Occasionally they would, but very little actually would come back to me. The form that I

used with Frank didn't happen to me very often. Now maybe that's a failing on my part, I don't know, but it didn't. Didn't happen very much. Again, the code complexity was so much that other people were put off.

Bochannek: It also sounds like you enjoyed writing the code, making it work, doing it well, and were less interested in directing a coterie of programmers to do this for you.

Doupnik: That's correct. I had my own problems in running groups for other reasons, and again it's just getting something done and this kind of stuff. Code wasn't the problem for me because I was fluent in all these things. It was just a question of time and energy to put into it and get something accomplished. If it were written in Swahili, I'd be just as happy <*laughs*> actually, even though I don't speak Swahili. I view it as a box, and somebody who really understands how the innards work, fine. Makes it work, and that's great. I wasn't interested in building an empire or running a large project because I had plenty of that in my own plate. Even in those circumstances where I am running groups, my interest is again in some goals, and the group is there to work with me until we get to those goals. Again, just because I don't like building empires, I continually do this to this day as a matter of fact.

Bochannek: Do you attend any conferences or lectures? Anything like that, related to Kermit? There's the Russia conference, and you already mentioned to me before the interview you didn't go.

Doupnik: Yes. There was one, only one that I went to. It was early on, on the east coast. Frank was there. It was the only time I ever met him, as a matter of fact. We had a large, very large meeting. I don't remember exactly the venue. This must be in the '80s sometime. I get up and made a schpeel on something, on what I was doing—and a very large audience. The thing that impressed me afterward was the—aside from the enthusiasm—the number of groupies that emerged from that. People wanted to reach out and touch and that quite frightened me because I can't—there's something not quite right with these people. It wasn't an ego appeal to me. I care less about that really, for one. The fact that these people reacted in that way was a bit odd. I used to talk with scientific audiences where it's anything but friendly. It's fierce <*laughs*> hand-to-hand combat, and I'm built for that and I love it. I'm used to that, I'm part of that environment, and this was quite different. It was almost pop in those days, and made me feel somewhat uncomfortable. Publicity was not my goal at all. Publicity has its use: to get people to become aware and so on. That's fine. But talking to audiences, no. I've talked with a large number of audiences all the time, and again, it's the same thing. It's not me <*laughs*>. It's it. It's it.

Bochannek: Who funded your work?

Doupnik: Me. <*laughs*> The Kermit project got no funding really, at least in my direction, at all. A couple of gifts came in. When we were doing the AT&T NetBIOS work, AT&T sent me a Unix PC machine. I think DEC may have given me a terminal way back when, and as I recall that's been it. No financial support for other things. Sometimes that's good, sometimes that's not so good. Just thinking how people are

motivated, creative people are motivated and things, money's usually not there, but boy does it <*laughs*> count in the end. You've got to eat and these things. But how to keep the creativity and all of the prestige and all that kind of stuff straight, money was not part of the mix.

Bochannek: What was the perception of your work within your department, within the university, even amongst friends and family?

Doupnik: Now, that's a good question because that really impacts what you can and can't do. My department initially liked it because they could use it. They didn't care about the rest of the involvement. My dean thought it was a really good idea that there was a little bit of publicity where it was at. Beyond that, they hated it because of the time involved, so it became a very serious obstacle to professional advancement. Even though I had taken the material in many cases and turned it into classroom material, you know, pieces of what I was doing—and so I simply told them that I was really doing more good in my estimation among more people by using this Kermit as a tool for them than I was by publishing scientific papers. I had to do both, but that was my stated purpose at the time, and I feel to the present that it was important for me to feel good that I was doing something constructive. I was going to continue, and they could do whatever they liked about it, and so they did whatever they liked about it. Life was very, very difficult for a really long time.

Bochannek: What about friends and family? Did you have support for this venture?

Doupnik: Only curiosity. <*laughs*> I don't have a regular family, so I don't have that particular problem. No, people are curious only because they see you working so hard on something. They hear little anecdotes and so on, but they don't look on the insides. That's true with even technology that I'm working on today. They don't care about the insides; it's just a thing, right? But you know, my being bounced around is an interest to a personal side of things.

Bochannek: Are there any specific stories you recall from users, from people who contacted you and said that they've used MS-DOS Kermit, and how it solved a particular need for them?

Doupnik: Only one comes to mind. I usually push those things aside in my own thinking, but there was a group doing a project flying on the space station. They came and said they wanted to use this MS-DOS program. I said, "Yes. Fine. No problem." They required some fixing up to tailor to their environment, which I did. Fine. They were most pleased, so it's probably still up there as far as I know, and that's been about the only real thing that's come across that way. Other people make comments, "Oh yes. We like it." "Oh, we hate it," which is fine, and I just toss those.

Bochannek: Is it your interest in space matters why this sticks in your memory?

Doupnik: No. Not really. It's because some group had a need that was quite serious. It was a means to an end for them. Just a piece of the puzzle, and they said, "We really are going to depend on this piece for these kinds of things." That's what sticks in my mind.

Bochannek: Why do you think MS-DOS Kermit was successful?

Doupnik: I think it just happened to fill certain needs. The terminal emulation was a big portion of that. File transfer was less because there were other file transfer protocols, and it was free. It seemed to have a reservoir of support people behind it, so you can ask questions. People were growing into the PC environment at that stage, so it was filling a bit of a vacuum that isn't there today, and the same thing is true again today. One of the subjects I'm dealing with currently is trying to take venture-provided material, which is a closed system and it only does certain things—and combine it with open source, which is a bit helter-skelter to do something more useful than how to make these things coexist, but keep both options going and things. I think in the Kermit days, we didn't have the commercial offerings, but we had one. We had one. I'll mention just as a way of anecdotal information again, there was a terminal emulator that came out of Attachmate, WRQ [WRQ, Inc.], and they were doing quite well. Today, the Attachmate group owns Novell, and I work with the Attachmate group. That WRQ is still there and things, and it's a commercial product so they keep it going.

Bochannek: We talked about some of the technical turning points in the project for you, implementing the different terminal emulation character support and so forth. We also talked about how MS-DOS Kermit ended with the advent of especially Windows 95. Any other turning points you can think of?

Doupnik: No, I don't. Getting in the communications protocols was very interesting, significant to my way of thinking. Getting modern terminal emulation end so I can do the various character sets and all these kinds of things was, again, a major do that was spread out over so much time, it wasn't a nugget. Those were the basic points of doing that stuff.

Bochannek: Are there any specific choices that you made, either technical or non-technical, that you regret?

Doupnik: Oh, I don't know. Probably some. I think the shortcoming was not doing long-range planning. One could do a deeper analysis and construct better systems, more flexible, longer-lasting rather than trying to work on something that can be done this month. That's a design error. That's strategic error on my part. I wasn't thinking in those terms, but looking back that's what should have been done. Now, I learned quite a bit of computer science, self-taught. You think about things to do it, which paid off very handsomely over time because I do know a fair amount about the subject. I hadn't thought at the time I was doing this to think in those terms and design well. MS-DOS Kermit has reached end-of-life. It can't easily be picked up and refurbished and brought into the modern world. Too much work; it's not good, and

it's not documented well enough. Documentation is the second shortcoming, which is something that we collectively should have done to identify the key features and explain them in simple English so that they can have a lifetime longer than a particular product. We should have done that, and that's particularly true because we're educational, we're university. That's our job.

Bochannek: It also sounds like this is a common issue with many non-commercial software offerings.

Doupnik: It is. It's worse today than I've seen it for some time, and people will create something, very few comments, slapped together, and issue it to the public, and then they walk away. I see this all the time, and it's too bad. Their ideas are not understood; their product is thought of as shoddy after a short period of time. It doesn't do what it needs to, and it gets lost <*laughs*>. It's really a complete waste because people can't articulate the key ideas and other things that allow the technology to move, to live in its own right. They had invented writing basically. That's really what it comes down—they hadn't invented writing that conveys ideas in abstract form, and it's tough. Technical people don't like doing that. Now I'm a professor and my life depends on doing that, and so I learned to abstract and distill and so forth, but it's a lot of work to do that and it's a complicated thing, and people don't do that.

Bochannek: Was there a decision you had to make during the project you felt was very difficult? Not necessarily technical; maybe just a direction to take, or that you really had to <*inaudible*>?

Doupnik: Nothing I can really think of. Clearly, Frank and I went through some ups and downs—technical difficulties, differing views on things—but nothing stands out that I really didn't want to get into, that I felt distasteful about. Some portions were quite a challenge for me to do, so it took a while to get around to actually doing a good job and seeing eye-to-eye on things. I don't know what Frank's view is looking back at his side of the equations, but compromises he had to make <*laughs*> because he couldn't convince me to move. I have no idea. No. I feel inwardly good. I think it was a successful project because we wanted it to be successful. No one was in it for any glory that I can think of.

Bochannek: Was there a particular accomplishment in the project you feel most proud of where you say, "Doing this is something you are very proud to have accomplished?"

Doupnik: Not really. The whole. The product as a whole is as close as I could come to it because that's the functional unit. That's what people see. Internally, no. I'm so used to facing technical challenges, puzzles, and overcoming them. Very few have a long lifetime of saying, "Oh boy, that was really a terrible problem and I got there. I'm really pleased." I do. I go through this many times of the course of a year that that knowledge fades. It's only when I can externalize it and say, "This is how things could be that we don't have now, so pay attention because things could be much better," that I get a feeling of accomplishment.

Bochannek: What was the most enjoyable part about doing this work? You said you like the puzzle-solving. What about the people? What about actually using the product? What do you enjoy the most?

Doupnik: Well, I actually use it. I enjoy the product. I'm using it more than anything else. I enjoy working with technically competent people. No, they don't have to be experts, but they can articulate things, they have common sense, and I'm a science person so interaction is the name of the game. That's where my rewards come from. It's peer review in simple kinds of terms. Other than that, no, it's driven by self-generated goals. I know this can be done; I think it's a good idea that we do it, so if nobody else does it, I'll do it and show you how to do that kind of thing.

Bochannek: You mentioned already sometimes the way the community of Kermit developers, Kermit users communicated, much of the newsgroup—there's the info Kermit list, the Kermit Digest, there are a number of different ways to do this. What did that community feel like to you? Was there conflict? Was it—

Doupnik: Oh, yes, there was conflict. Frank probably explained a couple of cases on Usenet when we had some real battles with external parties, which I won't go into.

Bochannek: If you want to tell the story.

Doupnik: I'm trying to remember all the details, but there was a person who had a product similar to what we had in some way. It was ZMODEM, and I can't remember the guy's name right now. Anyway, he had a strong personality and a strong way of doing things, and the question is would Kermit be able to do some of the same features? Could they be merged together and so on? Well, he and Frank did not get along *<laughs>.* Boy, oh boy. It was very nasty on Usenet, which is not a good place to have a battle on these kinds of things. That was an outstanding case of things simply not working, and he went off of it and huffed and did his own things. Mostly what I saw were a few very knowledgeable people who could make incisive comments, a large body of people who were making lesser comments because they were just experiencing this, that, and they thought they'd pass along the ideas and they're being kind by doing so, and then a vacuum. The people who were willing to risk making a comment are far and few between, and so we had this low level and very few really, really strong comments that came up, which is not great. If we had more modern communications and publicity and so on, there would be more of that, but that's about the way most systems work—a few strong voices, a few brave people, and then the masses who would keep quiet, which is too bad.

Bochannek: Talk a little bit about Frank da Cruz and the folks at Columbia and how you interact with them. You said you only ever met him once.

Doupnik: Yes. Met him once. Everything else was done with email, and it was quite interesting because we had a good relationship. Frank and I got along quite well. Not perfect, by any means. We had roles we played. He was in charge, and I was in charge of my area, but there's little else much I could say *<laughs>.* Compromise. We had the same general view of the world. That worked very well. I probably think that we would not get along if we were in the same room together for long periods of time *<laughs>.* Two completely different personalities. There's nothing personal. It was all object-based. That is almost all object-based rather than personal-based kind of thing, and it worked well. I have great respect for him. It was a pleasant time.

Bochannek: The ultimate goal that you both had of sharing this useful tool with everybody is the same goal. The impression that I get is the ideology behind it might be slightly different, but you both come out of an academic environment and it's just the natural thing to do—is to share the knowledge and share the tools.

Doupnik: That's right. Yes. That is correct. I think Frank—because he had to raise money to keep himself fed—was more interested in the project side of things; it's publicity and standing because he's had lots of battles with Columbia. I had fought and handled my own battles in a somewhat different way on things, but not seeking commercial support at any point in the game.

Bochannek: Is there anything that you would like to add? Any particular stories you'd like to tell? I think we made it through my list.

Doupnik: No. I think the morals are in the discussion we've already had. How people can work together if they choose, how they can externalize things that they have learned, and assimilate similar things from other people. Technology can be complicated and it gets lost if you don't simplify it to, say, make it visible. The pleasures of working in a cooperative environment—compromise and battles. In the end, achieving something that alone would be much more difficult—and people don't realize that there is satisfaction in that. Many, many, many years ago I was involved with a much larger project that was to put the first satellite in orbit, and working on the vehicle to do that, the program. The Russians got there first, as we all know, but the first vehicles were made in the US on this thing are now in the Smithsonian. I always point to people, I say, "Yes, I helped make that particular piece of gear. That piece of aluminum.". It's pleasure in being part of something good. That's it. I'm not being creative. It's just, I was part of that and I feel very good about that. That's about the only satisfaction you really ever get out of these things. I don't look back on it as bad times; I look at it as a very expensive and personal time. We really have to do better about these kinds of things. I look at academic institutions as being unable to deal with many creative activities that are spontaneous. Long history of that. I don't want to say wrong things, so I think I better stop *<laughs>* there. I'll get in really big trouble. The willingness of people to take on something significant and explore—the technique of exploring to some sort of conclusions is some skill that many people don't have; most people don't have, but they could have and we need to teach them a bit of how to do that.

They become a little more self-sufficient but also more creative and have a good time and these things, and I'm trying to do that now <*inaudible*> things.

Bochannek: That's it. I think that's all I have, so thank you very much for your time. Really appreciate it.

Doupnik: Okay. Very good.

END OF INTERVIEW

