Oral History Interview with Dennis Shea, Dave Brown, and Mary Haley
February 27, 2019
Interviewer: Laura Hoff
Transcribed by Cyns Nelson

Note: The interviewer is identified by her initials; narrators' remarks are distinguished by their first names, placed at the start of comments. Conversational sounds and verbal outputs that are not words are placed within parentheses; peripheral and editorial notes appear in brackets; words spoken with emphasis are placed in italics.

[00:00:00]

LH: I'm Laura Hoff, I'm the NCAR [pronounced like "en-car"] archivist, and I'm here at the Mesa Lab on February 27th, 2019. I'm here with Dave Brown, Mary Haley, and Dennis Shea. And we're going to be talking, today, a little bit about the NCAR Command Language and its history.

So I'll get started with just a simple question: where are you all from? And how did you come to be at NCAR? We can start with Dennis.

[00:00:27]

Dennis: Well, I originally came to NCAR in April of 1972. Came from CSU, Atmospheric Science Department, originally to work on a satellite project for two years, and ended up staying—to make a long story short. I worked as a—what is now known as an associate scientist, and more-or-less progressed up the ladder within that classification, for many years. And I worked on both science and on computer-related things.

LH: And they were part of CGD?

Dennis: I was part of CGD. CGD was not CGD, back then.

LH: Right.

Dennis: Just like we described off camera—or off tape (chuckling)—all these acronyms change. It used to be Atmospheric Analysis and Prediction. And prior to that it was probably something else. So, yeah. I've been in the science division all along.

LH: The whole time. How about you, Dave. Where are you from? How'd you end up here? (Laughter.)

[00:01:46]

Dave: Well I have, kind of, a circuitous career, I guess. I graduated from Stanford, with a degree in English, in '67, I think it was. And after that I was—I guess I kind of lived an alternative lifestyle for a while. (Chuckles.) I got into construction; I was doing building, as a house framer

first, and then I actually was—I was in the Carpenters Union in Denver, and went through the whole apprenticeship program. So I was pretty much into that.

But I started having—then I kind of got into my—the work kind of dried up after a while, so I was trying to do my *own* job, be my own contractor and do jobs for other people. I was basically getting sick of sawdust. I was developing allergies, and I wasn't making that much money, so I knew I had to do something else. My son, actually, kind of got me into computers, because he bought a—well, I got him an Atari 800XL for Christmas one year, and he really got into it. I mean, you know, he had been wanting it for a couple of years, and I finally got it for him. And then I got into it, too. So I was studying it, studying how it worked.

[00:03:37]

So, I did that for a while. And then, after about a year of that—and I was living on a low-budget existence, I guess. So, I had a friend—Jeff Haemer—who quizzed me about my computer-book [?] knowledge, to that point. And he said, "Well, maybe I can help. I have this guy who needs someone to teach Fortran, at Front Range Community College. And, the thing about it is that anybody who has real experience won't do it, because it's such low pay," and, whatever. So I ended up doing that; I spent 18 hours a day, for a while, on an Apple computer trying to figure out what the heck was going on, you know. (Laughs.) And I kind of just stayed ahead of my students. And I did that for a couple of semesters.

Then I actually—through another friend, I got an actual job at Precision Visuals, if you remember that. It's a while ago. So I worked there for a while, for a couple years, and was working in device drivers. That was a graphics company, so I was working on translating the final codes to pen plotters [?], and that sort of stuff.

Anyway, I got laid off from that job. But I got another job where I was working indirectly for Lofton Henderson who, apparently, had NCAR connections. And I think I got a good recommendation from him. And that was how I eventually got a job at NCAR, working for the new NCL project. Mary can comment on that, because she didn't vote for me. (Big laughter.)

Mary: I'm a little fuzzy on that! (More laughter from all.) That was 1992, right?

Dave: 1992, yes. I could go on, but I'll stop there.

[00:06:06]

LH: Okay. And how about you, Mary?

Mary: Yeah. So, I'm from New Mexico, I got a master's in mathematics from New Mexico Tech, in Socorro. Actually working at Mission Research Corporation for two years, for an astrophysicist. And he said—when he found out I was moving to Boulder, because my husband's job had moved here—he said, "You've got to get a job at NCAR." He had, apparently, been an intern here. So I said, "Okay."

I came in '91. I was here for about three months, and then I applied for a job and started with the Scientific Visualization Group, in 1991, under Bob Lackman. My first role was helping people with NCAR Graphics, which is what it was called back then, which is writing C and Fortran programs to do graphics. And somebody was going on vacation, apparently—Adrianne Middleton was going on vacation for three weeks, and they needed somebody right then and there to do consulting. (Chuckles.) So, I pretty much got hired right away.

LH: That's amazing. And then you—so you were there when Dave was hired.

Mary: Yeah. I was on his interview committee.

LH: And you had been working on visualization before that.

[00:07:17]

Dennis: Well I used NCAR Graphics quite a bit. So, NCAR Graphics—correct me if I'm wrong—I mean, there is a lot of history there. NCL is part of the evolution of that history, from when NCAR first started, let's say in the mid '60s. I mean, the building was here in '67, but it had started on campus. There was a culture, within the computing division, to provide software to enable scientists to do their jobs. And so, people like Dave Robertson, Tom Wright, a few other people, started developing NCAR Graphics, which is a package that can be called from Fortran, to do graphics.

That actually became, I would say—I had nothing to do with it—a package that was used worldwide by people who used Fortran, which was the—what do you call it—lingua franca of science. And so it got very, very well known. And it did great graphics. And I think you've got to mention Dave Kennison on any of these histories. Because Dave Kennison developed a lot of the more used packages within that. But, you know, it was an evolutionary thing, not a revolutionary thing.

[00:09:05]

So, then, my understanding was that, uh, used to be, the environment was big mainframes. And then you would submit a card deck, and that would be Fortran, and then within the Fortran you could generate computer graphics. And then they would have other devices like the dicomed [sounds like "di-ko-met"] to put out microfiche. Or you could do it—some other typed up pen plotter, things like that.

Mary: Little slides.

Dennis: Yeah. And so, my understanding was that NCL was—with the evolving technologies—NCL was to be a continuation of that, such that you wouldn't be tied to a mainframe doing batch jobs. Rather, you could sit and maybe enter commands. So Fortran is a compiled language; NCL, MATLAB, IDL, these other, are interpreted languages. We don't need to know the details, but *my* understanding was that NCL was to become something which will be built upon NCAR Graphics, to accommodate, let's say, the evolving new culture of interactive types of things.

And NCL was—again, please correct me—a "read-and-plot." Read data, plot data; with not much computation in between. And so, I think that that's when Ethan Alpert was hired, and Dave. And then Mary, you started working somewhere along—

Mary: Well so, Ethan Alpert and I were hired within three days of each other. And he was the original developer on the NCL.

[00:11:19]

Dave: And he came directly from C.U.

Mary: Yeah. He started off as a student project called GLOVe. And actually, there's a guy named Brian, on my floor, that remembers that project.

Dave: Really.

Mary: Yeah. They used to do these C.U. student semester projects. And so, NCL started from somebody who was involved in that student project.

Dave: And also Jeff Boote, can't forget him.

Mary: Yeah. Jeff Boote, he came in a little bit later. Ethan Alpert was the original.

Dave: Right. And they wrote—Ethan and Jeff together wrote the, kind of, the initial framework of NCL, I would say.

LH: And how did that, kind of, develop from there? Were you working with scientists? Were you working with external users, to kind of make it more—

[00:12:20]

Mary: So, initially NCL was developed—started development in 1992. And I think, around that time, because it was still not quite being used yet. We had an NCAR Graphics conference, in '92. Then we started getting people ready for the idea that NCL was coming out. NCL was officially released in 1995. I think that was the time that we had the second conference, right around '95 and '96, where we introduced people to NCL. We also introduced them to the underlying object-oriented programming (background chuckling) paradigm that was built on top of that. And I'm sure Dennis will have something to say about that.

So that was kind of our rollout of NCL. And I'll hand it off to Dennis, because that didn't go so well! (Light laughter.) That conference was kind of an eye opener.

Dave: And I can remember that, because I gave the talk on the object-oriented stuff. Whoa. Nobody liked that. I wasn't very—I didn't feel—very competent as a public speaker, in the first place. I had never really done that, much, at all, ever, before that conference. And so, that was kind of my first: "Ewww! I've got to stand in front of people and tell them stuff?" So, yeah.

Mary: It wasn't you as a speaker; it was just, the new object-oriented way of doing things was not user friendly. It wasn't obvious, from the get go, to somebody who wasn't a programmer. Because we were trying to sell this to scientists who are not software engineers, saying, "Well, we don't really understand this object-oriented interface. And it's not very friendly for us to use." It was complicated.

LH: And a big shift for them, right? In how they would work with the—

[00:14:06]

Dennis: Yeah. It was a different way of thinking about things. So, given a retrospective point of view, I would say: NCAR Graphics is Fortran and C. But mainly designed to be called from Fortran. And, it's more sequential type of thinking. I think that Ethan came from a computer science background. And at that time, the idea of object-oriented thinking was evolving. And I think it was a great—given hindsight—a great design decision to put an object-oriented layer on top of NCAR Graphics. So these objects could go down and interact with the underlying, nitty-gritty code.

The problem, from my perspective—because I attended that conference—was that a) the object-oriented stuff was completely new to all of the science people. And the main seminar room—my recollection—was 95-percent filled with people paid for by then SCD, Scientific Computing Division, to come here to be exposed to the new NCL.

The problem was that—like, when Dave gave his talk—this was completely new, if you will, out of left field. And the problem with the—even me, after a while, *even* me (background laughter)—how these objects—so, there would be a tickmark object, contour object, and how—and a map object, blah-blah. And how these objects interacted with one another was so foreign to the way people were used to thinking, that people were very frustrated at this first talk. I specifically remember going to Bob Lackman—who was head of the group, with Ethan, and Dave, and Jeff Boote, who was the other speaker at this particular conference—and I said, "Bob, this is a disaster. You have to give some examples of how this new object-oriented stuff is working." And, to Bob's credit, he came up the next day and actually had some examples of how this might work.

[00:17:19]

Unfortunately, I think a lot of damage had been done in the psyche of the audience in saying: "Whoa. We're just gonna go back to using the old Fortran." And, what's the best programming language? The one you're most familiar with. So people just do that as the fallback. To my knowledge—because this will eventually come out—not one person that I knew used NCL.

Mary: At that point.

Dennis: At that point. And, really, for—I'm gonna say—a year later—as this will come out—but, would you say that my assessment was correct?

Mary: It's hard to remember. But yeah, I do know that they were not happy, the way it was implemented then.

Dennis: I think that the assumption was that the science people would become programmers and put a substantial amount of time into learning how object-oriented thinking would be a benefit. And I think that was a problem; because most science people, they're not programming all the time. They go off and do their stuff and then they come back, and it's like: "How did that tickmark thing work? And interact with the other thing?" (Light laughing in background.) And so that became—that was an enormous psychological stumbling block. My assessment.

[00:19:05]

Dave: But then it was Mary that saved the day—

[Cross talk.]

Dennis: That comes out a little bit later.

Mary: Yeah, '97 I think, is when you and I started our collaboration.

Dennis: So, where the collaboration between CGD and SCD started was—and this is completely different—in 1995, the climate modeling section at NCAR decided to use NetCDF as the output format of the model. Prior to that, it had been Cray binary output. We had a person, Gloria Williamson, whose full-time job was to put out technical notes describing what each model run was doing, and the names of the files, and things like that. And the thing about NetCDF, that turned the tide: it's what they call self-describing. So, you could look at the header information and figure out what type of model it was, what was included, blah-blah. That type of thing.

So in 1995, completely independent of NCL, the modeling group said: "We are going to have NetCDF as the output format." That was another excellent decision. But then we also had, on the Crays, a Fortran-based processor called the CCM, Community Climate Model post processor. And it was a very large Fortran; a Fortran code that used NCAR Graphics as the graphical output. But, the Fortran had lots of Cray extensions, non-standard, so that they could use some of the hardware that was available on the Crays to make it fast.

[00:21:22]

And further, it was not user friendly. And, what happened was: the Climate Modeling Section at NCAR—who was really the parent of the Community Climate Model—said, "We need another tool to process the output data." And they didn't want people to necessarily always have to come to NCAR to use the Crays, which is what used to have to happen—literally, fly here and use it, and so forth.

And so, Climate Modeling Section, CGD, said: "We're gonna have a competition." And the competition was: something that could be used on non-Crays—Crays *and* non-Crays—that could be used to post process the data. And there was only one hard rule: it had to be able to handle

NetCDF, because now all of the model output was going to be in NetCDF. And then, there was a person by the name—there were two people, one Rick Wolski; I don't know if you remember that name? (Background "No.") He, unfortunately, had cancer. And he died. Then there was Lawrence Buja, who took over the CCM post processor. And he put together a list of the ten most common things done by the CCM post processor.

[00:23:14]

And so, the rules of the contest were: it had to be able to handle NetCDF; and it was suggested, but not required, that people make a presentation on how their tool would be able to address one or two of the most common things done by the CCM post processor. And so, the four tools that made it, that were nominated, were: MATLAB—Tim Hoar made the presentation—IDL—I think Gokhan from the oceanography group, and maybe another person co-joined him for IDL. Both MATLAB and IDL were commercial products. And there was a little bias against that, because there was some concern within our division that if Climate Modeling people were trying to train people to use the model, if they were gonna put out something that was in a commercial product, would they say: "Well, you pay for it." Okay, this was something that was sitting in the background.

The third tool that was nominated was Yorick, which was a product of Lawrence Livermore Labs, and it had a strong support or constituency within the Climate Modeling Section, the core group model developer. And I think it was Byron Boville and Phil Rasch—and maybe Jim Hack—made the actual presentation, made the case for Yorick. And Yorick had: a) it handled NetCDF, b) it was lightning fast, and c) it had at best mediocre graphics, but the guy who was developing that, said that he would really enhance that if it was chosen.

[00:25:22]

Mary: But I want to ask; but one of the requirements, not just being able to handle NetCDF, what were the graphics? That was a big part of it.

Dennis: Oh yeah. No, it had to be able to put out graphics.

Mary: Yeah. I mean, that's what they mean by "post processing," is that you're [cross talk] reading in these NetCDF files [cross talk]—

Dennis: The main output were the files that then were subsequently used by one of these other products, or put out graphics.

Dave: So I'm wondering: wasn't CDAT in the running? At that time?

Dennis: No. CDAT was not nominated at all. No Python-based thing was nominated.

Mary: I don't think it even existed.

Dennis: CDAT existed.

Dave: Yeah, it did.

Dennis: '95 it was out there.

Dave: Yes, it was.

Dennis: I was not involved, even though I was in CGD. I was peripherally involved because I knew I was gonna end up using whatever it was that was being chosen. But from my perspective, I thought Yorick was gonna be the head-on favorite. Because it had such strong support within the Climate Modeling Section. And again, the two commercial tools—there was some reticence about them. Also, MATLAB for sure did not have a company-supported NetCDF. There was something that a guy in Australia wrote, which was written in MATLAB, that would handle NetCDF. So it wasn't quite smooth.

I don't remember very much about IDL, but I think IDL did have some level of NetCDF support built in with it. And Yorick was—the guy made it very NetCDF, I don't know, favorable. Whatever you want to use, the term.

[00:27:21]

Mary: Then of course, the fourth tool was [cross talk] NCL.

Dennis: Well this is where, more or less, I start getting involved, was: nobody in our division used NCL. Nobody. But Ethan Alpert came up to my office, and he stunned me by saying: "I'm gonna enter NCL into the contest." And I said, "Well, you know people don't like the object-oriented nature of the graphics." The graphics are outstanding, the quality. But getting there was a problem. And so, he said: "Well, we can try to address that." But he said, "What can I do that would really be a hook for the people in CGD?" Because we were gonna make the decision. And literally, we were sitting in the main seminar room, and all of these things were done over a week of—a class on MATLAB would be held, and then an Octave—they also mentioned Octave, which is a public domain version of MATLAB, but it wasn't very well developed at that time.

And so, let's say the MATLAB "slash" Octave presentation would be made, and then people were supposed to try to use that tool to do what they needed to do. And then IDL would come a week or two later, then Yorick. And then what happened was: NCL—and Ethan asked me, "What application would be the best?" And so, I said, "The most common thing done is to convert the Model Coordinate System"—which is what they call hybrid levels—"into pressure levels." Because that's where all the operational data from the weather service comes out, is on constant pressure surfaces. At least it used to be.

[00:29:38]

And so, I said: "Do the thing where you do vertical interpolation." And that is why V-I-N-T, H to P—that's the name of this function—is the only built-in function that handles metadata.

Mary: Right.

Dennis: That's how it started. And so, Ethan, a) he did the best presentation. He put some thought into it. And he really de-emphasized the O-O, object-oriented nature of the graphics, and he showed results. And what happened at the end of the, let's say, two-month process, a vote was taken—literally, people in the main seminar room—"What tool do you favor?" And overwhelming, hands went up in favor of NCL. Why? Number one: access to the developer, *developers*, to get stuff done for us. Because the vinh2p people were thinking: "We can get *all* of the stuff that's in the processor into NCL." And Ethan did a wonderful job on the presentation. And so, presentation's important.

And then—this was *my* view, kind of the key. And there's some history to this; I'm sure Mary and Dave get tired of me talking about history, but sometimes there's a reason why things are the way they are. And so, back in the 19—I'm gonna say 1980, plus and minus a couple of years, up until that point, prior to that point, SCD/CISL [sounds like "sis-el"] always provided the programmers for the model. And one of the—the division director at the time, Stu Patterson, pulled support, literally overnight. Stunned our division. And it became a thing within our division of: you can't trust SCD to follow through on anything.

[00:32:01]

So what happened was: after NCL had been chosen, then the division director of CGD, Maurice Blackmon, came into my office and he said—and you can take this any way you want; when a division director asks you if you would be willing to do something, it's not a question (background laughing) so much as a directive—and he said, "Would you be willing to make an evaluation on whether to accept"—I call it the Yay or Nay decision on NCL. And I said, "Why ask me?" Because our group was real data. And I said, "I have nothing to do with the model." And he said, "That's the point." He was a professor at University of Washington for a while, and he realized how little support people had, working on just general-science issues. And he said—because I had also helped people in our division do certain dirty, dog-work, data tasks—and he said, "That's why I want you to do it."

And that's when I started working with Mary and Dave. I mean, in some sense, my only knowledge of both of them was the '95 workshop, and then maybe seeing each other in the hall. I mean, I don't remember any interactions prior to that. I may have had some NCAR Graphics questions that somebody would answer.

[00:33:56]

So what happened was—this is why I always say, "Mary Haley saved NCL." (Background chuckling.) And that's not BS. Because, the very first thing was: how do we get around the stumbling block of the object-oriented interfaces? How do we get around that people have to know the plot manager, which always used to come in—and even though you'd say, "I told it to do this, and it won't!" (Background chuckling.) This plot manager was like, supposedly, a guardian angel. (Big laughter.) "You don't know what you're doing. We're going to ignore that." (Big laughter.) And that's the truth; there's something called the plot manager.

And so, (background laughing) I said: "This has to be addressed." And Mary Haley, who developed what are called the GSN—with the initials Getting Started with—

Dave and Mary: Using NCL.

Dennis: Yeah, right. So, what happened—these are what we call "high level" interfaces. So people basically put the data in, and then there's something over here, which, if you want, you can change certain things. But under the hood, Mary's code got around all these interactions between the different objects. It hid it. And that was the key to me—and there was a part two to this, too—me giving the "yay" to the division director, Maurice Blackmon.

[00:35:47]

The second part was: Mary and I—and this is when I realized how hard Mary Haley worked (background chuckling)—the model back then was a spherical harmonic-based model. And the Climate Modeling Section told me that if I didn't have spherical harmonics in there at some level, that they were gonna really have a lot of push back. And there also was several people telling me: "You can't trust SCD," based on the cutting of support, "that they'll follow through on supporting this tool."

So, Mary and I—it was during a summer—worked. We'd be on these VT100s, working out interfaces to put spherical harmonics into NCL. So there are functions—you can call this little thing, and it does a lot of stuff. So Mary and I did that. And based on those two things—hiding the object-oriented nature, and putting spherical harmonics in—that's when I told Maurice Blackmon "yay." NCL can do the job. But then—neither one of you were present—because of this history of, let's say, our division not trusting CISL on support, Maurice Blackmon insisted that we have a meeting with the then power structure of SCD: Bill Buzbee, Pete Peterson. Bill Buzbee, division director; Peter Peterson, associate director; Bernie O'Lear, deputy director; Paul Rotar, who was head of the Systems group; Dave Kitts, who was the main programmer in the Systems group. Don Middleton was there. *Maybe* Dave Kennison. I don't think so, but Don Middleton was there. And then, representing our division was: Maurice Blackmon—

[00:38:17]

Mary: I think it was also Bob Lackman, was—

Dennis: Bob Lackman was there. [Cross talk.] Yeah, okay, there *was* somebody else. Right, there was. Bob Lackman was there. Then there was, from our division, Maurice Blackmon, Lawrence Buja and me. And Maurice wanted assurance from the CISL/SCD people that they would support, if you will, in perpetuity, NCL. They wouldn't pull the plug on it, okay. And, unfortunately, there's no paper to prove this. And really unfortunately, a number of the people that are there are no longer with us. So, as a result of that agreement, NCL was chosen as being—if you will—the official CCM post-processing tool. It was the only one that was gonna get support from our division, okay, at that point. And so, then Maurice Blackmon actually came up with money for a couple of years of support. And I wrote the job description for us hiring Sylvia Murphy.

And Sylvia Murphy came in, and her job was specifically to talk within our division, within the different sections, what they wanted to see within NCL, to allow them to do their jobs. And Sylvia also came up with the idea of what eventually became our applications page. In other words, here's a script, and here's what you get. What that did was: it allowed people to essentially download the script, and at least then they were off to a running start on trying to get something done.

[00:40:29]

But then—I mean, you guys can chime in—then what happened was: I taught, I *learned* NCL, really, interacting with Mary primarily. I ended up teaching classes within our division, holding them in the director's conference room. Like, ten people at a time. And I came up with exercises for people to do. And our lab sessions were: they called me up on the phone, when they got stuck, and I'd run up to somebody's office and help them out.

And then what happened was: Pete Peterson, the associate director, called me up—I'm gonna say, '97, '98, somewhere in that time frame—and he said, "I know that you have been teaching people in your division." And he said to me, he said: "I have five highly-paid people working on NCL, basically." And he said, "I need more." And, just being a CCM post processor. And he asked me if I would be willing to teach it to people outside of NCAR. And I said, "Well, you have to have approval from CGD. And how are the finances going to work?" And so, Maurice Blackmon, I think, Pete Peterson—I'm not sure if Bill Buzbee was literally in on this—but they worked out an agreement, because it would be part of CGD outreach; CISL outreach; our division would continue to pay my salary, regardless if I was doing something that was NCL oriented. And they would cover—*they*, CISL, would cover all my travel expenses.

Isn't that how it ended up working, to your knowledge?

Dave: I think so.

[00:42:50]

Mary: Yeah. That was right around—that was February, 2000, what he's referring to, the NCL workshops.

Dennis: Right. But there was preparation before all of that happening. And, the first workshops were taught by Sylvia and me. And then, moving forward, in 2005, there was a big layoff in our division. And the two people who were primarily tasked with NCL-focused activities were laid off. And then it became me doing that.

Dave: Well, was that when you got involved, then, Mary?

Mary: Yeah. It went on—the workshops went on hiatus in 2006. And then you and I started teaching them.

Dennis: And Mary and I taught 50 workshops.

Mary: Seventy-eight, actually. Seventy-eight workshops, and then you retired. And then I think we did three more. (Chuckles.) So, about 80, 81 workshops.

Dennis: Yeah, that's total. But you and I—

Mary: Oh! You and I together, probably about 50.

LH: What was the structure of the—were they weeklong workshops? Were they mostly held here?

[00:44:08]

Mary: It was a mix. So, we tried to do two local workshops at the corporate training center. We moved around. Before we had the corporate training center we had other locations that we did it. But that was our main location. And we would do two of those a year; usually one in the summer, and one in the winter. And then we tried to shoot for one at a university workshop. We started off with UCAR member universities, but then we opened it up to any university. And then sometimes we would get a fourth or a fifth one, maybe, internationally or at another university. So we were doing almost four to five, sometimes—the most we ever did in one year, I think, was six. Maybe seven.

And there were three—well, it was a week long, basically, when you add in travel and the amount of prep that you had to go into it.

Dennis: I think the mornings were, if you will, lecture; and then the afternoons were feet-on-the ground, interaction, sitting—right, like if you had something, "What is it you want to do?" Because we did ask all of the students to come with a specific objective—an *obtainable* objective, not doing their PhD while we're sitting at the—and some people wanted us to sit right there. And they were intense.

Mary: Yeah. We called it—we really tailored it to—we said, "Bring your own data." So the idea behind the workshop was: we would teach you how to use NCL. But then—it's usually about 16 to 20 people in the classroom—we will try to talk to each one of you individually to find out what your objectives are. Bring your data to the workshop—be ready to bring it—and then we will work with you to create NCL scripts and look at *your* data. So that way, you weren't giving canned exercises that they may not be interested in. You were giving them real things that—they were getting real work done in that workshop. And then by the time they left, they had something—hopefully—that was working, that they could take back with them.

[00:46:05]

Dennis: I think the key was—and this is where Dave was so important—was that there are a variety of data formats out there. Okay, the model was NetCDF, but there are other formats. HDF—there were several different flavors of HDF—and the real killer for a lot of people was something called GRIB, which is what the operational weather centers use. And I always say: The thing that—in the olden days—that was most difficult about data processing was to get it in,

so then you could do number crunching. People are very creative about doing number crunching, but bringing data in can be an incredibly tedious and difficult process.

And where Dave came in, was: Dave was—I'm gonna say—the I/O guru, and he wrote these interfaces that got down into handling GRIB and HDF. And a key to NCL is that their data structure is not an array, like you might have in Fortran, or C, or something like that. But it was actually a data structure, which handles more information than just the numbers, but it includes descriptive information. Well, it's temperature; and here's what the units are, and may have associated with it the coordinates that tell you were to plot all of this stuff. And how to isolate different areas. And so this part—I think, unfortunately, the graphics are the glitz: "Oh, that's pretty." But the guts are: getting the data in, so you can do what you need to do to get it to do the graphics, okay.

[00:48:12]

And I think that's where Dave—I've said it many times in the workshop, Mary can testify—with NCL you never have to worry about a data format. And that's because Dave developed the software that takes these disparate formats, and when the data is imported, it puts it into the same data structure; it makes it look like you're looking at a variable from a NetCDF file. So, this consistency of the data structure, across different formats, made it easier to also write computational routines, because you're always dealing with the same thing. Do you know what I mean? The same familiar territory. And I think that is very important. And so Mary's graphics routines, they—when you pass down an NCL array, or a data structure came down, these graphics routines are saying, well—remember, it's self-describing—"Tell me about yourself. What do these numbers pertain to?" "Oh, temperature." "Well, what units are you—" "Oh, yeah, degrees centigrade," or whatever it might be. And oh, "you want to plot over the US? Okay, I've got these coordinates; I can just do it for you." So, all of this tedious crap-ola is hidden from you. And I think that's a key that made NCL a success.

[00:49:55]

And then, I think the feet-on-the-ground interaction with the users; if you only look at the modeling stuff, it's actually—I always say, processing model data is pretty easy, because you get perfectly-written NetCDF files. The only problem with model data is: there's a lot of it. But that's a computer science issue, like how you can use different processors and whatever. But the real key is these applications. And from the workshops—you've all sat in these things—the spectrum of stuff done by people is enormous. And, as a result of these interactions, a lot of scientific utilities were developed to go between the data being read in, and the data being either saved out to a file or the graphics. And so then the middle part is the computations. And how you are deriving certain things to see if things are operating the way you think they should be working, and so forth. And so then the middle part, I think, largely a lot of those application software came out from interactions with users; feet on the ground, literally sitting next to people.

And if you hear the same request, different places, then you know it's something that people want.

Dave: (Softly.) And you put some effort into it.

[00:51:39]

Mary: We've always felt that that was kind of the key to the success of NCL, is being able to do those workshops. Watching how people are using your software, you go back and you improve the software. And in the next workshop, hopefully, you're able to use that new feature. And it's so incredibly important—when you're developing software—not to develop it in a bubble, without talking to the scientists directly, and watching them try to use your software. Because us software engineers don't know how they're thinking or what they need to do. So, I think every one of those workshops is like you're fine-tuning the software.

Dennis: And learning [cross talk] new stuff. I think one of the things—

Dave: But of course, the mailing lists also were a key part of that.

LH: The mailing lists?

Dave: Well—

Mary: [Cross talk] For people that couldn't attend workshops, they would send an email saying: "Well, I'm trying to calculate monthly anomalies," [cross talk] "and I need to be able to create this kind of"—

Dave: And the fact that we were pretty responsive about that, you know, probably—

Dennis: I think it was the key. I mean, in the big picture, the key is support. Developing a software and saying, "Well, this thing can do wonderful things, you're on your own." I think, in particular, at smaller universities, there's no infrastructure for support. Or very little of it. And I do think that NCL support was sometimes over the top; I mean, responding within an hour or two of somebody sending in a request. You're getting an answer, and you're getting not just *an* answer, but the right answer. That's the key.

Dave: Yeah.

Dennis: You knew you could get graphics or I/O support; or maybe computational support. And you're getting the right answer. And I think that's incredibly important.

[00:53:38]

Mary: And to be—when NCAR Graphics, back in the early '90s, originally we charged for support. We charged for NCAR Graphics, which was I think $750 dollars for a site license for a university. They got to pick one person at that university that could ask a question. So if anybody at that university had a question about the software, they had to go through their site representative, and that site representative could then contact us. So when we went to open source, with NCL, that model went out the window. We were no longer charging; we were no

longer requiring a single point of contact. So we basically opened it up: anybody that wants to ask a question.

It was a little overwhelming at first. But I think that it really resulted in a much better software.

Dennis: I think it ended up helping us focus where we should focus. But it also led to some levels of abuse. (Background chuckling.)

LH: That's sort of a double-edged sword, when you offer such comprehensive support, right? You get to kind of benefit from what the user needs are, but you also—

Dennis: I think it's appreciated a lot, too. I mean, I've had people—when I used to go to scientific conferences, you wear this thing, name saying "Dennis Shea." And I had people coming up to me; I'd be eating dinner: "I hear you're Dennis Shea." Or they'd see me in the hall, and they'd say: "Thanks," for the help. And so that always spurred me on—perhaps too much, but spurred me on. Because people appreciate it. And you know, people will say: "I don't know if this particular thing is being used." Well, if it works, and you've described it appropriately, then you're not going to get a question. People say: "Well that's the way it's supposed to work." You only get questions when people misuse it, or they don't want to spend the time learning how to do it.

So you know, support's kind of—it's important, but you can be taken advantage of, too.

[00:55:51]

LH: So we've touched on a couple of things that I think are unique about NCL. One is the support, the level of support. You mentioned the "any format," compatibility.

Dennis: Any Common Data Format.

LH: Any Common Data. That seems unique to me—as a non-expert—but that seems pretty amazing. Are there any other things that are unique to working at NCAR on NCL?

(Light chuckling.)

Dave: Well, working at NCAR has been pretty unique, for me. I don't know how to say it, exactly.

[00:56:30]

Mary: Well, I think the chance to work with people from all over the world—

Dave: Yes, that's very true.

Mary: —and the user, the email list that we have has really opened that up. So the—you know, we talk about the file I/O, and the computations and the graphics. Those are kind of the three

pillars of NCL, of what made it so popular. Sometimes people just used it for the graphics, or they might just use it for the file I/O. But all three of those things—or the computations—were equally powerful. So, I think, being able to work with people from all over the world was a real benefit of this job.

And we were able to keep metrics. I think from about 2010, we started being able to track, like, where NCL was being downloaded to. We didn't know *who* was downloading it. But we could see—it would give us a lat-lon location. So we started to see that it was being adopted—it was kind of funny, because of course the United States had a lot of users. And then in other countries it was sort of like on the coast—you would see these little dots showing up on the coast. And then they started kind of moving more inland. For example, in Africa, initially all those were coastal users; but then we've seen more users, now, more even spread, across the different countries. So that's been kind of exciting.

There's people that we've gotten to know over the years, that—we haven't met them in person, but we've kept up correspondence with them quite a bit.

[00:58:04]

Dennis: All of us—and in particular Mary and I—have traveled at the request of people. Like in Australia, the three of us went; Germany; Mary and I went to Korea.

Mary: Switzerland. France.

Dave: Switzerland, yeah.

Dennis: Switzerland, yeah. I mean, we've been all over the world together. A benefit—not just professional, but personal—was meeting people from all over the world. And I think that's something that was an unexpected benefit of ending up being involved heavily with NCL.

LH: And then, for more of the local workshops, you mentioned that there were some—in an email—that there was some work done to make sure that people from EPSCoR regions were able to be involved in the workshops, or to go to those places, to get traditionally underrepresented geographical areas in the sciences involved.

[00:59:15]

Mary: Yeah, correct. CISL put up money every year to have—you know, we would probably get about six to seven students, a year, that we would fly here for the local workshops. They had to fill out an application saying why the workshop was important to them, and what they hoped to get out of it. Usually, it wasn't that hard for them to get in. And then they had to be from one of the EPSCoR states, which are states that don't get as much benefit from scientific technology advancement as other states. And there's about 26 of them, actually, so it was not hard to find somebody from an EPSCoR state. So we would pay their fees. And also, if a university was in an EPSCoR state, then we would go there and teach the workshop there for free. Although I think most of our workshops, that we taught at U.S. universities, we always tried to make it free. I

think, as the budgets got more tight, then we would start asking for a little bit of—you know, make half a payment, or something like that.

But it worked out well. I think we managed to get a consistent number of people from EPSCoR states, and we also had the minority-serving institutions. Not as many; it was open to both, but we tended to get more from EPSCoR, just because there's a lot more of those.

So our last workshop was actually in Idaho. But it was not any of us, it was Bill Ladwig and Scott Pearse teaching the new stuff now, the VAPOR software, and more Python. So that was the first time a workshop, I think, has been held in Idaho. And I think that's an EPSCoR state.

[01:01:00]

LH: Before we get into the new stuff (laughs), I just want to mention that you all shared in the outstanding achievement award in 2005, for your work on NCL. I think you, maybe, put together the paperwork for that award?

Mary: Yeah.

LH: And that was just to recognize the evolution of NCL, right? Is there anything else you want to add, about that?

[Cross talk and laughter.]

Mary: It's interesting, because NCL really—in 2005, I don't think it had hit its stride, quite as much as it is now. Because that was—2005, ironically, was right before the workshops went on hiatus. So, at that point, I think there had been maybe 30—maybe 20 to 30 workshops and hadn't quite—I don't think, at that point, had done any international workshops. Because Sylvia—when you and Sylvia did the workshops, those were all university based. I don't think you traveled internationally with Sylvia.

Dennis: No, I did not travel internationally. We did do several labs in Washington, and then down in—I think you were down in New Orleans?

Mary: Yeah. Right before Katrina.

Dennis: So several *national* labs, but not internationally. I mean, when people came here, some of them took the workshop, and then some of them said: "You should come to our lab." And that's how some of these international workshops evolved. Wei taught, also, in China.

Mary: Yeah, a couple on his own. Because he taught them in Chinese. And so—yeah, there was no way.

Dennis: Our Chinese is a little (background laughter).

Mary: China is actually one of our biggest numbers of users. And when he taught the workshop—I think there were two workshops, and they each had 40 people in them. I don't know how he did it; because we barely do it with two of us and 20 people in the class, you know.

[01:03:06]

Dennis: I think the difference is that we did it with heavy interaction—four, if you will, required days a week, wasn't it? Of five hours: four hours in the afternoon. And then an optional day on Friday, in the mornings, for people who had residual questions and so forth. Little, even heavier one-on-one type of stuff.

And so, I mean for computer stuff, you can always just give a lecture, and say: "Here's how you do a do loop," a repetitive loop. And, "Here's an exercise, go ahead and do it." And you know what an answer is. But when you are asking people to bring their own data, that's when you learn—I call it a "mind numbing" spectrum of applications that people are doing. And that's why I think I/O is the key. Graphics are the glitz, and the computation are the guts.

Mary: But in the case of the Idaho workshop—because that software had been tailored to hurricanes—but when they got to Idaho—this was more Python—they're interested in looking at data over the mountains. So suddenly the developer—before they went to Idaho—suddenly had to tweak the computational routine, to account for mountains. And so that's the kind of thing that: interacting with the users directly, getting their data, working with them, has really made a difference in the software.

LH: Seeing all the different use cases. And so, what do you see as next steps? And what's the future of this type of work?

[01:04:59]

Mary: Well, we're all retired! (Big laughter.) Well, at least two of us are. Yeah, well, the next step is—I mentioned Python; when NCL was initially developed, in the '90s, Python *was* around. Python is another scripting language very similar to Python [NCL], but it was not being used for scientific purposes. It was really kind of new, and it wasn't tailored to the kind of work that we do here at NCAR. So it made sense that NCL came around. But, by the mid 2000's—around 2004, 2005—we started to see that Python was really being adapted by a worldwide community, not just in business.

Dave: I'll disagree a little bit with that, about it being developed for scientific. Because, I think CDAT was actually started way back, around the same time as NCL.

Mary: Well, it just wasn't as heavily used by the scientists.

Dave: Well, not here. Not at NCAR.

Mary: Right.

Dave: But at Lawrence Livermore, it probably was. I don't know.

Dennis: I think Los Alamos—not Lawrence Livermore, Los Alamos is where CDAT came out of. But I think that—I mean, off the record (background chuckling), or on the record I guess—I had people contacting me offline, saying—

Dave: They didn't like CDAT.

Dennis: They did *not* like CDAT. They were being forced—

Dave: I understand that.

Dennis: —because of contracts. Because DOE put so much money into it, they had to use it.

Dave: But I think that's later.

Mary: Yeah, that was later. I mean, '94, I don't think— [cross talk]. When NCL first started development, it wasn't—

Dave: Python was just coming into focus in that early '90s time, around the same time. I mean, Konrad Hinsen—the guy who developed the NetCDF reader for Python—that was written in the '94, '95 time frame, first.

[01:07:23]

Mary: It's weird. Because I just don't recall Python—

Dave: No, I'm not saying that it was on our radar. But it was happening.

Dennis: I mean, the fact that *nobody* mentioned Python when the contest came out, gives an indication of how widely it was used.

Mary: It may not have had graphics at that point. Matplotlib is what—kind of the de facto graphics that are used in Python today. And I just don't know if that was—

Dennis: I think they were based on MATLAB-style graphics. I think that's where the name kind of came out of it.

Mary: But so, long story short, it really started to pick up steam—at least for us—in the mid 2000s, and that's when Dave developed PyNIO [pronounced pie-knee-oh].

Dave: Right.

LH: And that was combining Python with the I/O? Is that—

Dave: Right, right.

LH: My rudimentary understanding! (Laughing.)

[01:08:18]

Dave: So, yeah. I took Konrad Hinsen's code and modified it to read kind of the guts of NCL's—instead of reading direct from the NetCDF library, it was reading from an NCL library that I kind of made out of NCL. So, yeah.

Dennis: I think that there was—since I know very little about Python—there was a library in Python called Numeric, and it was the predecessor to what is now NumPy [spoken as "num-pee"].

Mary: NumPy [spoken as "num-pie"].

Dennis: NumPy. There were issues with numeric—I mean, these are nitty-gritty, low-level stuff—but as I recall, they were changing the numeric underlying model, and so forth. And so, there was—in the very deep background of a lot of people—"I don't want to get involved with something where I'm gonna have to be changing things based on what some Python guru says should now be done." Okay. And so, that was something that was in the background. I think that—again, from somebody who knows almost nothing about Python—but I think NumPy, which became more the stable numerical underpinning of array operations in Python, appeared in about 2006. And so, that provided some level of stability, from my perspective. Prior to that, I think that there were some issues with some of the libraries, and so forth. But, don't take my word for it.

[01:10:23]

Dave: I mean, it's true. Numeric was the original thing, and I think that was developed—I believe—with Konrad Hinsen, Paul Dubois, those guys. But then, it's true, there was a—then there was a split, because the people at the Space Telescope, they're big Python users. And they developed their own version of it; they called it something else. And I forget what—

Mary: NumArray.

Dave: NumArray. Yeah, maybe something like that. Anyway, they developed their own version, and they were not compatible. And that's when Travis Oliphant eventually wrote NumPy and tried to unify them together again. That happened around 2005, like you say. Yeah.

Dennis: So, from a science perspective, the last thing you want to end up doing is worrying about these libraries and how they're interacting (background chuckling). "Will the thing I'm writing now—if I have to go back and redo it, do I have to start changing things?" Because, to me, that's a nonstarter for most science people. They might enjoy programming, but they don't want to *live* it. And that's the rub that comes in.

I think that I have a different perspective, and I obviously have a bias towards NCL. It's a focused library, and it's primarily—but not exclusively—focused on atmospheric, quote-unquote,

or climate-related things. I have an _____ [?] [sounds like "ocean"] library, by the way. I never put it out there.

[01:12:21]

I think that the reason why Python has, kind of has—and I'm gonna just say this—a cult that follows it is: Python is present in many different fields. Because Python got its reputation, I think, as a gluing language. In other words, you could implement something in some other language from within the Python environment. And, so, let's say SQL—which nobody in our field uses, really—there's a big SQL-Python library. And there probably are several other things. So things that use SQL, they can say: "Well, I can use Python to get at the data, and maybe use another library to process it. Something I like."

So, I'm gonna say: Python has attained a certain level of ubiquity. But you know, within any particular area, I'm not sure that you can say that the libraries available within that area are particularly well vetted. And my take is: before I retired, there was something that said, basically, "NCL is wrong because it doesn't agree with this Python library." I'm just gonna say, it wasn't NCL that was wrong. But the idea is, that because it came out of the Python environment, it's correct. And that pissed me off. (Background chuckling.) I mean, it's—one of the people within the, let's say, the SCD community quotes—I don't know who the guru of Python is, I forget his name—

Dave: Guido Van Rossum.

Dennis: Guido "whatever." It was this comment from him, and it was about how wonderful Python is. And what went through my mind was: this sounds like David Koresh of Waco. (Background laughing.) Or Jerry Jones of— (Background chuckling.)

[01:15:00]

Mary: Well so, yeah. They are trying to—it definitely has a cult, but they are trying to address that. They're trying to—the people that use Python are very fond of it, and tend to promote it very heavily.

Dave: Remember that time at that SciPy, when Paul Dubois—I think it was—said, "This should never have happened!" He was talking about—after your talk—

Mary: Yeah, well, and open-source software—what he's talking about is: We introduced something called PyNGL [pronounced “pingle”] which was our Python interface to the NCL graphics. And at that time, matplotlib also existed, so it was somewhat seen as a competition. I introduced PyNGL at the SciPy conference—was it in 2006?

Dave: I think it was before that; I think it was 2000—it was the same—because I heard that "what's his name," the matplotlib guy, just before he died—

Mary: John Hunter?

Dave: John Hunter, gave a talk—another talk, much later, in 2012 or somewhere. He was talking about the beginning of his talk, and he mentioned your talk about PyNGL as—I think it happened at the same time. I mean, he had been developing it, but he gave one of his first major talks on it, at that same SciPy meeting.

[01:16:28]

Mary: The question was: why are you developing PyNGL, when matplotlib is already out there? I had just given my talk; I said, "Any questions?" And Paul stood up and said, "This is outrageous! It should have never happened!" Of course, in today's diversity and inclusion (laughter), maybe we wouldn't ask the question that way, nowadays. (All laughing.) Because we're all supposed to be respectful. You know, the point was: Why are you duplicating efforts? And it was because PyNGL was really—because it was based on NCL graphics, the graphics were really well done. I mean, they'd been in development for 30-plus years. They had their own following, because that interface was very well known to the NCL users. So we were taking the NCL-like interface, but giving them the option to use Python as their programming language, but still being able to carry that graphic interface with them, and understand it, and be able to tweak. Because the thing about NCL is: you can really tweak the graphics to get every little part. You want your tick mark to be a certain length, or you want your colors to be just so; or you want your label bar to be a certain height or width; and I need my fonts to be a certain size. You could tweak just about any part of the plot, which made the graphics publication quality; really clean, sharp looking graphics. So that was sort of our point for PyNGL.

[Cross talk.]

LH: —and allowed users to kind of continue with the same experience they were used to, right? [Agreement from others.]

Dennis: What's the best programming language? The one you're most used to. And the implementation done by, primarily, Dave I think—

Mary: Yeah. I mean, Dave deserves the credit for that interface.

Dennis: Is that the—you could map NCL graphics, and there's an enormous—that application page has 500 examples of—

Mary: Over a thousand.

[01:18:34]

Dennis: Over a thousand examples of graphics, some of them incredibly complicated. And so, what you could do is, look at the NCL script—and NCL, a resource, which is an option—let's say the width of it, its viewport [?], width—so it goes: let's say, I'll just say "res" at, with the "at" sign, vpWidthF equals zero point eight. Okay? And to map that into Python, or PyNGL, what you do is: you go res at "period," not an "at" sign, "period," width. And that's it. I mean— [cross talk]. And so, it's almost one-to-one. The difference is that NCL graphics are aware of this

metadata structure that's coming down that has the descriptive of what the variable is, and so forth. And in a Python library, the user has to explicitly say: "This is temperature," and so forth. But, the really dirty part—that's just the nuisance—the real dirty part of how you do these graphics—and I think, Mary, you'd agree—of all the questions we get in NCL, two-thirds are graphics related: "How can I get my tick mark to be 45-degrees angle [spoken with squeaky voice] and have it as aqua blue." (Background chuckling.) You know what I mean? That type of thing.

[01:20:18]

So, the thing that—Dave's mapping of NCL graphics into a library that can be used from within the Python-osphere is almost one-to-one. So all of those graphics that are on the NCL—

Dave: I can't take credit for that. That was Fred Clare, a lot of it—

Dennis: Oh! Was it Fred Clare? [Cross talk.] Okay, I didn't realize that.

Dave: Yeah. And Mary; Mary's done a lot of it, too.

Dennis: Well, it's all teamwork. [Cross talk.] So, the graphics—and like, the person asks you, we have an enormous user base. We don't want them to have to learn a new graphics paradigm, and so forth. We put out really good graphics, and people *like* it, and like the looks of it. And that's what we're trying to do; we're not trying to make *your* graphics our graphics. People can choose yours, if you want to. And that, to my knowledge, is part of the culture of Python, is that you can have all these different things under it. Why should we have to be shoehorned into yours? And so, I think that what, let's say, the graphics people within NCL have done—I think that was outstanding. And people can, almost right away, if you can just get the data in, in Python, people that are familiar with NCL graphics can just, essentially, highlight an NCL script and change all the "ats" to "periods." [Cross talk.] (Laughter.)

Dave: It's not quite that simple. (Laughter.)

Dennis: I mean, in a gross sense, though. Let's say there are a hundred resources or things you have to do. You're still going to be using 90 of them.

[01:22:18]

Mary: But what that gives you is that: by doing that, we're enabling people to use Python. And Python opens up a world—for example, we have a lot of people that say, "Well we want more statistic routines in NCL." Well, we just don't have the expertise, in our group, or the budget to hire a person just to do statistics. But R, which is a well-known package for statistics, does have a Python interface. So now, if somebody's using Python, they can use R to do their statistics; they can use PyNGL to do graphics if they want. They could choose to use matplotlib for the graphics; they can use PyNIO for the file I/O. So it's sort of a mix-and-match.

Dave: You know, then it has web-based stuff, so you can put stuff right on the—use your http, all that kind of stuff.

Dennis: It's designed for doing stuff onto the web; on the other hand, for R—and here's one of my rubs—is that R puts out their data structures, and I am low-level functional in R. So whenever I would put a statistics routine in NCL, I checked it against R. And some of those examples: I say, "Here is the R script that I used, and here is what NCL produces." So, you have confidence to do that. The problem with R is that—and I sent this in an email to Mary, a long time ago—R, they write out their own data structures in R. And so, the reason why there are so many different versions of that R interface is: they're always adapting to changes within the R environment. And so, it's not like, "Oh, this thing just worked perfect." You find out there's a problem. And within R, you can—depending on what you, how you want your output—you can get a lot of information. And I don't know, but I'd be stunned if that type of stuff was necessarily available in the R—the interface to R—from Python.

[01:24:35]

And one of the problems comes in with R—and I think I mentioned this to you, I have a book on R, that's how I kind of learned enough to just become basically functional. They said: one of the things that R core group—and it's like the Python core group—if you get something developed by the core groups in any of these, it's top quality. The documentation is relatively consistent from one application to another. You're familiar with it. The problem with R is that so much of the software, I think, my recollection is: this book says there are 15,000 libraries, R libraries, or functions out there. The problem is: when they're developed by all these different people, they all have different levels of how things are—how the interfaces work; how they view their software. And you know, while it works for *my* data, doesn't necessarily mean it's going to work for *all* data. And so this is a problem that the R community is trying to address—that R core group. And I think Python may be even gonna have to address this; and it's a much wider spectrum. R is focused on statistics, and Python is on everything.

Dave: Everything.

Dennis: But, you know, just because it's there, doesn't mean it's done well. And that's the rub. [Pause.]

Do you agree? I mean, you guys know Python; I don't.

[01:26:19]

Mary: I forget what the original question was, but we were just talking about why Python has—it's just kind of picked up steam—

Dennis: It's a gluing language, because it's—

Mary: —in the atmospheric languages. It also has hooks for doing—better than NCL does—for handling large data, for doing parallelism. It's just a cleaner language in general; has more

features. So, people tend to like it—and they're learning it. That's what graduate students—if they're learning any language in college, scientific language, they're probably gonna be learning Python. It used to be, MATLAB was the—and it probably still is, to some extent—but more people are moving to Python, because it's open-source, it's free. If you learn MATLAB, it's not a free software. So then if you leave the university and go to work for a particular company, and they don't have MATLAB, then suddenly you don't have your favorite language available.

Dennis: But I think MATLAB learned—IDL did not learn—to do what Microsoft has done in business software. So if you're a student, you get the software at some low level. And the most important thing in computer science, from my point of view, is building familiarity. And so, MATLAB makes a student version available. And then you end up—as I understand it, every year after you graduate—or within—you still have to pay, like, $50 bucks a year, maybe, to gain some more access; but you have MATLAB. And so, if you have MATLAB, and you're familiar with it, that's what you use.

Most science people are gonna use whatever they're most familiar with. And NCL—like some of the other tools that are more discipline-oriented in our field—is something called GrADS. And then there's something called Ferret, which is more focused on oceanography, and it's out of PNNL—Pacific Northwest Labs, or something along those lines. But they're very focused, on those things.

[01:28:45]

Mary: Well, so what the—I think what you're trying to talk about, to some extent, is that the support for NCL, of course, is very tight. Because we develop it here, we have full control over the software. So if somebody asks a question, we can look at the software and say: "Yeah, that is a problem. You're right." Or, "We can fix this." Something like Python or R, that's being developed worldwide: somebody has a question on package X,Y,Z, well where do they go to get help? And they're at their desk; they don't know who to go to. That package may not be supported anymore.

But NCL was always supported here, and so it's kind of a tough decision to say: "Okay, we're gonna adopt Python as the language that we develop all of these new things on top of." So now we're gonna use Python as the language; we've stopped developing the file I/O, because we lost Dave Brown (background laughing) when he retired. He was our file I/O guru. But Python now has a package called xarray that's being developed here—well, I think it's being supported here, very heavily supported here at NCAR. It gives us the same metadata model that Dennis talked about. Being able to read in a data file and have all that information available to you; that's what xarray is to Python, is what Dave Brown and Ethan Alpert's NetCDF model was to NCL.

[01:30:14]

Dennis: Trying to pass that down, let's say, into an R application. R application's not going to be aware of this.

Mary: Well, no. But see, xarray is NumPy under the hood. So everything is done—all scientific computations, if they're done correctly, they're done with NumPy. There's no other way to do it. So some, xarray, you can easily get a NumPy array out of it. That way, you can—anything that handles NumPy will be able to handle anything coming out of xarray.

So xarray, right now, has PyNIO as its back engine for NetCDF and GRIB, and HDF—

Dave: Not NetCDF—

Mary: Oh, you're right.

Dave: Just GRIB.

Mary: Just for GRIB, And maybe HDF-EOS. But they—you konw, that's something—so now we're able to say, "Well, we're going to hand that off." We don't have the staff, anymore, to support the file I/O, unfortunately. So we're depending on xarray, and the people developing that, to—for example, I think the UK Met Office is now producing a GRIB reader that's pretty powerful, in Python. And that's gonna be added to xarray—or is already in xarray.

[01:31:26]

Dennis: Have you tried using it?

Mary: Yeah! I have, actually. [Cross talk.] I wrote a script—

Dennis: How do they keep up with all the different operational centers?

Mary: I don't know. I just—

Dennis: I'll bet you they don't do that; that was what Dave did. (Background chuckling.) That's what was the tedious, time-consuming—because the first, what, 128 IDs are guaranteed by the World Meteorological [WMO] zero through 127, right?

Dave: Right.

Dennis: And then, everything beyond that is up to the operational centers to put out. And you don't know what those numbers pertain to. What does ID 232 from operational center in Indonesia pertain to? You get numbers, but then you have to literally look, on the Web or on a piece of paper, and say what that is. And that's where Dave stepped in and provided that extra information. And it's tedious and time-consuming to do that. Wouldn't you say?

Dave: Well, yeah, it was tedious and time-consuming. But I'm hoping that they have—I mean, I'm thinking that they probably are closer to ECMWF, and WMO, and so they get those updates, maybe, more.

Dennis: Well. let's hope so.

Dave: They don't have to go ask for them; they get 'em sent to them. That's what I'm thinking.

[01:32:53]

Mary: So right now, you know, somebody's using Python; they bring you an xarray, and they can say: "I want to use the PyNIO GRIB reader, or I want to use the cfgrib," I think it's called. I forget the exact name of it; I think it's cfgrib. EC code under the hood.

Dave: EC codes, yeah.

Mary: And that's kind of the idea around Python, is that—

Dave: So now—let me ask you—do they organize things in arrays, like—

Mary: Yeah.

Dave: Yes, they do.

Mary: Well, xarray—in fact—

Dave: Xarray does that.

Mary: Yeah. The idea being that: xarray saw how well NCL handled its data model, and allowed you to bring in these different data formats and have it look like the same thing to a user. Well xarray does that for Python. They allow you to read in NetCDF or GRIB. I don't know where they are with HDF support. But NetCDF and GRIB are kind of the two biggest ones that we need to worry about. My testing seemed to indicate that they're pretty well there; I mean, cfgrib is still fairly new, and somewhat in early stages.

So that's where we are now. We're transitioning our software to Python. We're letting go of the file I/O; we're keeping it for now, we're keeping it in maintenance mode, just in case. Because we're not sure that they're completely there with GRIB, yet. The graphics, we know, there's nothing like it in Python. So we're going to keep that going for a while. Matplotlib still exists, and we actually have some very good matplotlib programmers here at NCAR: Ryan May is one of them, over at Unidata, actually. And he's offered to help in any way that he can. Because there are some things you can do in PyNGL that you cannot do in matplotlib graphics. I think curly vectors—I'm not sure about that, but that's the one I use.

[01:34:52]

And then, of course, the computations. That's where the big whole is, right now, in Python. We don't have—all the work that Dennis and I and the team have done to put the scientific calculations into NCL; those are not available in Python, in *any* package. Some of them are, but not—there's some really high-quality, tailored programs that we put in, functions that we put into NCL that need to be ported to Python. So that's the top priority right now, is, figuring out—because we have thousands of functions; maybe hundreds that need to be moved over. And keep

NCL going for a while, because it's not like we're yanking the rug out, saying, "We're done with NCL." But we just can't—the staffing levels, and the budgets just don't allow us to continue supporting—NCL is a *huge* package. It requires a lot of support. And with Dave Brown gone, the file I/O can't continue. We basically have three people developing it right now; with me gone, it will be three people. They're going to put their efforts into computations and the graphics.

So that's sort of where we are: Pulling things over to Python, and transitioning—trying to help our users, because as Dennis pointed out, we have hundreds of examples where they can go download an NCL script. That thing has been one of the most popular features that NCL users—if you look at our web statistics, because it's all web-based, you'll see that's one of the top things they hit, is that applications page. And we would love to have equivalent written in Python. There would be nothing like it.

LH: So, some of it is just getting people used to this new way of doing—

[01:36:46]

Mary: Yeah. And you know, NCL's gonna live forever. Honestly, there are still people using NCAR Graphics. Which was written in the mid '60s. And I still have people writing to me—just last week, NCAR Graphics question. I said: "I'm retiring! Sorry." (Laughter.) You know. And NCL's not—it's still there, it's open source. It's on the web; people can download it. There's no reason why it shouldn't still work.

Dave: It's on GitHub, so people can, supposedly, develop it, too. Right?

Mary: Oh, yeah. We're moving to open development. So that means: "Here you go." We're gonna try to get people to help us develop this stuff. And that's kind of how Python is. Python is very much an open-development community. NCL has been more closed development. We're gonna take components of NCL, the computations, put it into a new library, and say: "This is open developed." Of course, we have to make sure that anything coming into the package is still vetted, somehow. That's a big challenge for us. I still say "us," even though I'm gonna be done in two days! (Laughter.)

Dave: Someone will have to be the curator of figuring out how to accept things.

Mary: Yeah.

LH: This has been so interesting. I've learned a lot! (Laughter.) I don't want to keep you all for too much longer, but I just want to say thank you, all, for your time. And just see if there's anything else you want to add? Before we wrap up?

[01:38:13]

Dennis: Well, I said to Mary, a couple of days ago: my opinion is that NCL has been the longest running, most successful program between a couple of divisions, than any other project.

Mary: Between CSD—well, now, CISL—and CGD.

Dennis: I mean, CISL can say: "Well, we provided hardware." And actually, there has been a culture of providing software like NCAR Graphics. But I would say that NCL has been—I mean, it's been going on for a long time. And it has a really good reputation. I think that it's a really good example of how software—because I couldn't do what these guys have done. Maybe they couldn't do what I have done. And somehow or other, putting it into something that is fairly intelligible, and useful to the community. I mean, there are some applications within NCL—I don't foresee anybody else doing it, because I always feel: if you create it, you support it. You don't just throw it out there and then go have a latte; do you know what I mean? Space-time spectra; the MJO CLIVAR type stuff.

Dave: Yeah.

Dennis: Who's gonna do that?

Dave: I don't know. (Chuckles.)

Dennis: I mean, you can make software, but if somebody has a question—who's gonna answer it?

Dave: Yeah.

Mary: That's what the key to the NCL workshops were: that we had a scientist and a software engineer, and then maybe another software engineer and another scientist helping out in the lab. So you were able to cover all the bases. Because I couldn't answer the science questions. Dave could—

Dennis: We'd be in the class, and I'd—let's say, somebody asked me something computer-y, I'd go: "Mary, or Dave, this is your question, over here." And they can get it addressed by somebody who *knows* what they're talking about, as opposed to somebody who *thinks* they know what they're talking about. (Background laughter.) Not the same thing.

LH: Right. It is a very impressive collaboration. [Agreement from interviewees.]

[01:40:36] [Audio stops.]