An Interview with

STEPHEN WAMPLER

OH 202

Conducted by David S. Cargo

on

25 July 1990

Flagstaff, AZ

Charles Babbage Institute
Center for the History of Information Processing
University of Minnesota, Minneapolis

Stephen Wampler Interview
25 July 1990

Abstract

Wampler discusses his work on the development of the ICON programming language in the late 1970s at the University of Arizona under Ralph Griswold. Wampler focuses on the implementation of Version 3 of ICON written in the C programming language.

STEPHEN WAMPLER INTERVIEW

DATE: July 25, 1990 INTERVIEWER: David S. Cargo

LOCATION: Flagstaff, AZ

CARGO: Steve, I was wondering if you could set a context by giving some of the background you had when you first started getting involved with the Icon Project.

WAMPLER: I started graduate school at the University of Arizona under Ralph Griswold beginning a master's program in 1972, switching to a Ph.D. in 1976. At that time Ralph had been my advisor and helped me, and I had had him for a few courses on the SNOBOL programming language. When I started my Ph.D. work, Icon was essentially at the development stage in Version 1. They had finished SL5, the precursor to Icon at that time. I had had some C experience. I started porting the Version 2 of Icon to the CDC 6400 that was on campus there. That's essentially where I started on the Icon Project.

CARGO: So you were a graduate student...

WAMPLER: A graduate student, with Ralph as my advisor in the Ph.D. program. At that time I was looking for a project, and I originally was interested in portability. I started on the Icon with the idea of looking at it and porting it to other machines. At that time Version 2 of Icon actually was written in FORTRAN, specifically RATFOR, and produced FORTRAN as output rather than its own interpreter as it is today. I don't know if I can add much else to their beginnings of it at that time.

CARGO: The decision to try to do a C implementation of Icon, how was that decision made, and what were some of the factors that went into it?

WAMPLER: The department, I guess about 1977, 1978, finally got a PDP 1/70 running UNIX and the first C compiler on campus. There were two of us involved in the project besides Ralph at that time (two students who were involved) - Cary Coutant and myself. We were both using the PDP for work and decided that C was a natural

language to go to, and the feeling was that C, if we wanted to head for a UNIX environment, would be a much more natural development system than FORTRAN. We were not aiming for portability so much at the time, other than the fact that since we were writing under UNIX on a PDP, C was the language of choice to do that with. Cary did the majority of the compiler port at that time. We actually had a compiler, not an interpreter going, and wrote a compiler for the machine. Partway through that work he realized that there would not be much difficulty in writing an interpreter as well. And so he wrote an interpreter shortly after the compiler was finished. Most of my time was spent on the Icon run-time system and trying to find some research out of the Icon Project.

CARGO: You were looking for a research topic then?

WAMPLER: Yes, if you're a graduate student, that's your primary motive in anything, and Icon being new, it certainly had a lot of areas in it. My own interests were primarily the language features more so than the implementation. The implementation was seen more as a vehicle to try things out with than it was an end-all. Consequently, we made some decisions back then that made sense in the context of running on a PDP 1/70 that don't make sense anymore, and that some of the later versions of Icon have corrected.

CARGO: When you said, "We made decisions," who was the *we*?

WAMPLER: Primarily Cary and myself. Ralph wasn't too concerned about how we did things as long as we got results. We were both trying to fit a fairly complex language into a small address space. A lot of our decisions were based upon the fact that you only had 64k bytes of address in which to put data, and so on.

CARGO: Nowadays that seems quite quaint.

WAMPLER: Yes, it does. [laugh] We don't worry about that so much anymore.

CARGO: What kinds of control was Ralph putting on the kind of work that you were doing?

WAMPLER: He, I felt, was concentrating on overall implementation design, making sure that we didn't do anything too insane, but primarily on the language itself - looking at the features, making sure that things were consistent with an overall view, keeping things from either getting too far out or too machine specific within the language, and those types of ideas. His interests were a level up from implementation, and much more interested in making a language coherent and a suitable research vehicle for further language study.

CARGO: When you and Cary were looking at adding language features, would you give proposals to him, or say, these were the directions that you were looking at?

WAMPLER: Yes, almost always when we came up with an idea it would be discussed with Ralph - the three of us and whoever else was there at the time. Tim Korb was still around for the very beginning of that, for example. He was the primary factor in Version 1, the primary implementor of Version 1 of Icon. And whoever knew Icon would be brought into the discussion on language features in that way. Occasionally Cary or I would implement something and then show it just to show that it worked, and so on. But usually there was a discussion of "Does this meet kind of the overall philosophy of the language? Is it something that is interesting enough to add to the language, or is it something that can be done other ways, equally easily?"

I wish I could remember more of the types of things Cary was working on at that time. I think he did a lot of work on string scanning, if I remember correctly, and just designing that, deciding that string scanning is an operator, and how to implement it were the types of things that were being done. Later on, after Cary left to go into industry and I was trying to get my Ph.D. done, I spent more time on co-expressions and the control structures of the language - looking at what could be added. At that time Ralph had to spend a lot of time looking at things, since that's again a little bit higher level work than the interpreter itself, and had a very strong influence on the features that were coming in.

CARGO: When you came up with the C version I noticed that it was mentioned in the Icon Newsletter way back in February 22, 1980 (I guess that was issue number 3). Did you see much demand for that C version from people getting the newsletter, people wanting to get copies of the implementation?

WAMPLER: I don't know. I never saw any correspondence on the Newsletter; Ralph was handling that. I suspect so; it seemed to me like once the C version came out we began to see a lot of interest in it. The FORTRAN version was just a little awkward in using, in bringing up. It was a fairly large project. The C version, having to fit on a PDP 11, was very small and tight, and as long as you had a UNIX machine it was fairly easy to bring up. I think that's really when the interest started to take off. The real take-off to me was really with Version 5 though, when Bill Mitchell came in on the project. He was much better at preaching Icon, and he would go to a UCC conference and some places like that with Icon buttons, and so on. That really seemed to help it take off.

CARGO: Can you tell me about PI?

WAMPLER: When I completed my Ph.D. and came up here I was looking for a research project (up here to Northern Arizona University). One of the things that I always felt was, because we had never spent much time on the implementation, it was not as efficient as it could be. And we had never really looked at the implementation in itself as what can we do to make it a strong implementation? Icon was an interpretive language, and it was a slow language even if you compiled it, for a variety of reasons. I felt, and still feel, that a large cost of that is in its flexibility. Like most things, when you get a lot of flexibility, you pay a price sometimes in performance. I felt that if one wanted to one could design an implementation of Icon that was considerably more efficient, which I call Production Icon, or PI for short. Unfortunately, I never got the work done; we don't have any graduate students up here to do any programming, and I ran out of time on my grant to do much. But I think the current work on the C compiler being done by Ken Walker would be what I would view as an extension. It's a completely different direction, but it's the same idea where it's the implementation that's slowing it down, not the language itself.

If we can predict in Icon how data types are being used, then we can get rid of the use of descriptors in a large part of the language. A descriptor essentially doubles the size of the data item, so every time you move one around, you are doing twice as much work as you would otherwise. So all data work is at least half again as slow as it would be in a language like C where you only move the data object and not the description of it around. I think that's what we are going to see. Production Icon was attempting to predict and eliminate the use of descriptors as much as possible in

the language, so that we could just move data. If you needed an integer you could just assign an integer or a register and not also have to move a tag field telling it that it was an integer around and so on. Profiles of Icon tended to show that most of the cost was fairly uniformly spread out throughout the language. That would point a finger at something that's throughout the language, such as descriptors. Achieving any sort of significant speed up in Icon overall would have to result in getting an implementation where you would not need descriptors as much as a simple implementation does. That's where PI was. I did some preliminary work on it, but ran out of grant money from the university before I could do much programming.

CARGO: When you graduated from the University of Arizona with your Ph.D. were you interested in staying at the University of Arizona? Was that related to the Icon Project in any way? In fact, you are still in the neighborhood.

WAMPLER: No, that's more pragmatic. Yes, I have always liked the University of Arizona. I certainly have enjoyed the work with Ralph and would like to continue that. However, I am not primarily a researcher; I tend to much prefer teaching and do the research secondary. The University of Arizona is a research-oriented school and that's the primary focus in terms of promotion and advancement. So I wanted a school that would allow me to primarily focus on teaching and make the criteria for which I get promotions in teaching, and do the research as a secondary aspect. The reason for staying close was the fact my wife's family is in Tucson and she wanted to stay near her family, and so we looked. This turned out to be an ideal location for me, although it is hard to do research, I have discovered, if you don't have graduate students to carry out all the slave work. But it lets me do the teaching, keeps me close enough to the University of Arizona, and I can stay involved in the Icon Project in that way.

CARGO: How would you characterize your involvement with the Icon Project since you came up here?

WAMPLER: Oh, primarily on the periphery. When I first came up here I got a little grant to do some work on helping bring up Version 5 of Icon, which was for VAX 780, and finally we had enough memory and horsepower to start doing more things on the language. I did a little bit of work on Version 5, and since then I have just been involved in discussions on the network, occasionally looking at things that are being done at the University of Arizona, and helping Ralph host this workshop every few years. I am not at all involved anymore in the implementation and the

design features that are going on now, other than as the person to bounce ideas off of.

CARGO: Do you communicate pretty regularly over the network now?

CARGO: Fairly regularly. It comes and goes in spurts, as will happen. It seems like some graduate students down there will take an idea and start to work on it. At that time there is some discussion in the Icon Project over features and how to do things. Once that's pretty well worked out you tend not to hear from the graduate students because they're too busy trying to get things actually running to do much discussion. And then finally, when things are done, sometimes there's a hindsight-type discussion that goes on about features.

CARGO: Would you say that that's greatly facilitated by everybody having access to the Internet for handling electronic mail?

WAMPLER: From my view it is, yes, because I would not be involved if I could not get involved electronically through Internet and through the Icon Project mailings, because I am just too far away to wander in and see what's going on. I don't know how much impact it really has on the work going on at the University of Arizona though, because there's enough people down there that they have their own critical mass to do things. Probably the area where the network has been the greatest influence has been in terms of increasing the number of ports of Icon on new machines, and that people who are far away and want to bring it up on a Cray, say, are able to get fairly quick help from the University in terms of "things aren't working, or how might I proceed" type things. I think that's really where the network has been the greatest help.

CARGO: Also, accessing the source?

WAMPLER: Oh, and grabbing the source, yes. Every time they come out with a new version I can get it up here fairly quickly and get it running on the machines that we have. I don't think we have any machines up here that they don't have down there, so I am not doing much work to do that usually. That's right, the source and having it available, although I think the biggest access has been on the bulletin board that they have run down there for the

PC stuff. I think that's really done very well bringing it out.

CARGO: Can you tell me a little bit about how Icon Project grew while you were at the U of A? It sound like at the very beginning it was primarily just two students and Ralph, and I got the impression that at the end it had grown quite a bit.

WAMPLER: Well, I think that is right. Icon wasn't originally designed as a language that was going to be released and become extremely popular; that was not its goal. It's goal was to be a language to let people study language design, both in how you implement Icon and in the features it provides, and in using it to do other things. There have been a number of projects that have grown out of Icon - multi-paradigm programming languages, for example, that people have taken from Icon, used Icon as a basis to develop, and things like that. Originally it was primarily something to do research in, and the first work grew out naturally of SL5, which had a few people working on it. Dave Hansen was involved; Tim Korb; Ralph, of course. I think Walt Hansen was involved near the end of SL5, and the realization that SL5 was just too big and too complex to get much done with, although it was a great language for doing research. If you are trying to get research you don't want to spend forever waiting on things to get done.

That group, with Ralph at the lead, changed directions and started the first design of Icon as a simpler refinement of SL5, making it a more practical tool - losing some of the flexibility but gaining a lot of practical abilities in the language as well. Then Tim finished and Walt Hansen left, and during that same period they were finishing up Cary Coutant and I came on as graduate students. We did not really enjoy the FORTRAN implementation so much and were excited about learning UNIX, so we started working on developing it that way. Again, no idea of really "this is going to be a language that's going to be widely spread" at all.

Cary Coutant left; I completed my Ph.D., and there were a few people coming in at that time, one of whom was Bill Mitchell - one of these people who needs very little sleep. You'd find him on until five in the morning very often with a great deal of energy and interest in Icon, and he was responsible for most of the Version 5 port and for at that point saying, "We need to do more than just do research with it. It's getting something that can be very popular." And so, along with Version 5, came a realization from Ralph that there's going to have to be something set up to support the

people who want to use it.

Not all of them are researchers and willing to live with the sorts of things a person doing research might live with, and the interest would be different in terms of what they want out of the language in various ways. And that's really when Icon Project started.

I had left before really the term "Icon Project" started being used. And it really was a response (you may get corrections from Ralph on this). I view it as a response to the fact that it was discovered there were a lot of people out there who liked Icon and wanted to use it, and the University was not set up for helping those people. It was set up to get research done and help the students that are there. Ralph formed the Icon Project as a way to help this need of people who were not that familiar with Icon or not willing to go in and tinker with it so heavily, or who wanted to port it to new machines, and so on. And I would guess that really started around 1982 when we started to see the Icon Project form. After that it became clear that there were really two classes of people - those that were really very interested in the language and how it works internally, which needed the help of Icon Project, and people who were just using it and wanted to be able discuss ideas, pass programs around, complain about features and suggest alternatives. And so Icon grew, came out of that as a separate thing. That was essentially not so much run out of the University of Arizona, but a network entity, whereas the Icon Project has always been very closely tied to the University with some people like myself and others scattered around outside of it.

CARGO: Again, that connection was possible because everybody had access to Internet mail.

WAMPLER: I think so. If it had not been through the network mail and so on the Icon Project would be exclusively, I think, just the people at the University of Arizona, and I think they would be very busy at that. It's a drain on resources to handle that, particularly when you have graduate students who have to make their primary goal getting a degree and getting out, and not necessarily answering questions or providing support.

CARGO: When did you graduate from the U of A?

WAMPLER: I graduated in December of 1981.

CARGO: That was with your Ph.D.?

WAMPLER: Yes, that was with a Ph.D.

CARGO: The first time it was clear you were located here at NAU was mentioned in Icon Newsletter #15, which is quite a ways, time wise.

WAMPLER: Yes, I have been here a long time, since January of 1982. It's a different environment. There's not the high play with Icon now; I don't do research on it, but I enjoy teaching, so I am enjoying that a lot. In case it's not down yet, David, let me tell you, the principal players in Icon were Ralph, of course, Dave Hanson, and Tim Korb early on, and then Cary Coutant and myself, and then Bill Mitchell. And I think the next major player is Ken Walker. There have been a lot of other people doing some really, really interesting things - Janalee O'Bagy, Kelvin Nilsen, and so on. But the next big player is Ken Walker and his work on type inference and developing an efficient compiler for the language, which I think will be the next nice thing to have as a faster version of Icon.

CARGO: When Ralph is recruiting people to work on the Icon Project, do you think a lot of the graduate students are familiar with Icon first and see the U of A as a good opportunity, or do you think many of them are just applying at grad schools and then Ralph picks them up?

WAMPLER: I don't know. Certainly for those of us that were there while Icon was being evolved we didn't know of it before. Bill Mitchell knew of Icon. He was at the University of North Carolina as an undergraduate and knew of Icon and had done some work there before coming here, so he was essentially recruited on that. I really don't know these days how that's being done. I know that there are people who are specifically interested in the University of Arizona because Icon is there. How Ralph is recruiting - I suspect that he also gets a very large number of people who take his classes and discover, a) how easy it is to work with him and, b) what fun it is to play with a language like Icon, that are getting into it. I would guess most of the students are still coming that way - discovering while they are

there.

CARGO: Was Bill Mitchell a graduate student then?

WAMPLER: He came in as a graduate student, got caught up in doing things, getting things running and got his master's, I think. He may have had his master's before he came; I don't remember. But I think he finally decided to go into industry and is still in Tucson and still involved somewhat on the Icon Project, but his work keeps him very busy.

CARGO: Would you say that most of the graduate students you are aware of are in a Ph.D. program as opposed to a master's program?

WAMPLER: Yes, I think that's very true. With all the courses you have to take on a master's, you don's see very many master's students able to really get into a topic deeply enough to do a lot of research in it. You will see some preliminary work that may indicate that they would be a good person to get into the Ph.D. program and throw research at, but usually they're too busy with course work to do a lot of work. You can't spend seven or eight hours in one day developing something and trying it out if you have got a test coming up, the next day. So almost exclusively the researchers, the people working on Icon, will be Ph.D. students.

CARGO: Have you seen a lot of collateral development - people in industry now who have picked up Icon from the network? You talked about how the University had the Icon Project to provide support; do you see much of that?

WAMPLER: A fair amount. AT&T up near Chicago has done quite a bit. Since I am not involved so much I have to go back in the past to answer that. When I was at the University of Arizona I would regularly get questions from people working at Tektronics, say, who were using Icon for simulations, and so on. So I would say there was a fair amount of that. Again, many of those people were not so much concerned with the research aspect of Icon, but just simply how they could use it - that they found it easier to use than their own simulation language, or easier to use than PL/I. I can still remember one person aghast that his 1200 line PL/I program became, I think, something like 57

lines of Icon, since many of the things he was doing, having to write himself in PL/I, were already done for him in Icon. But there was a lot of interest, from my point of view. Every few months we would hear from somebody wanting to know something about it, and since the language was brand new I thought that was pretty good. These days, someone who is there now would have to answer that.

CARGO: I am asking about history, after all.

WAMPLER: One interesting thing. We have a person here who got his degree from Oregon State who had actually been at the U of A for a while, and he has gone another step. He has taken Icon and used the goal-directed evaluation and some of the concept of co-expressions and streams such as were in Segue, a language Ralph developed out of Icon, to develop his own language. And I think that's the first completely new language that used Icon as a basis. There have been other modifications. Somebody in France, John LaCarne, tried to write also a PI. That was for Pascal Icon, where he started with Pascal and tried to add Icon features into it. But the language G is the first one I have seen that took the fundamental ideas in Icon and tried to expound on them into a totally different language, and that was kind of interesting to see.

CARGO: Who was it that did that research?

WAMPLER: John Placer. He was actually a faculty member here at NAU now. It's a different approach. His language does not look like Icon. You have to look at it a little while to realize that what's going on underneath is some of the ideas that were out of Icon. And I found that really interesting to see that. To me, that means a language has matured in some way - if you begin to see the fundamental ideas being used other places.

CARGO: Have you seen any other examples of this collateral development?

WAMPLER: No, I haven't. Not recently. Again, I have to go back to the late 1970s and early 1980s. The work in France was one of the big things that was being done. There was a lot of interest on Icon and still is - I think the University of Nice in France where John LeCarne is. And there has been a few correspondences from people in

Japan. The most interesting thing I have seen on the influence of Icon was I got a call from an IBM salesman in England once. He was in an absolute panic because some university had asked for bids on machines, and one of the requirements was the machine run Icon, and this salesman had never heard of Icon. And so he was desperately trying to find out if there was one for an IBM mainframe; at that time there was not. But that told me that Icon had spread pretty well. It was showing up in England and being used to bid machines.

CARGO: Can you estimate a date for that?

WAMPLER: I would say about 1985. I had been up here three or four years by then. And it *was* amusing. I could not figure out why someone from IBM in England was calling me. I don't know where he got my name, but he wanted to know if there was an Icon for the IBM mainframes.

TAPE 1/SIDE 2

CARGO: You said that you had some grant money to work on PI. I was wondering if you could tell me more about whose grant it was and how you went about getting it.

WAMPLER: Okay. Let me start by saying NAU is a fairly small school, oriented towards teaching - no graduate students in computer science, for example. It's hard to get grants up here. The first grant was actually out of the University of Arizona in that Ralph had a grant that could fund a faculty member at another school to do some research. And that was essentially working with Bill Mitchell on Version 5 of Icon - doing some other work. Then I applied for a local grant at the university here since it was difficult for the university to get funding - particularly in field like computer science. They also provided local grants and I had one of those to do PI. It was funded through what is called the Organized Research Grant up here at Northern Arizona University. It was a little bit of a problem in that I felt I needed half-time to do any significant work without graduate students, and they could only fund me for quarter-time. But the department was willing to help give me some extra time to work on it, and so on. The work was fun. As I have said, I just did not get a lot of programming done on it. I learned this from Ralph that if you are doing research you should be able to show that it works. It's not really sufficient just to say, "Oh, I know that it will work,"

but you have to be able to implement it, and PI never really reached that point. I keep wanting to dust if off, although I think Ken Walker is going to pretty well beat it into the ground with his work. He will certainly do everything I wanted to have done.

CARGO: Were you aware of where Ralph was getting his funding?

WAMPLER: All or most of it was from the NSF, yes - and several large grants over the years. For quite a long period of time, I think, he has been able to get significant funding for the Icon work. It's difficult to get funding for something like the Icon Project where you are not doing research. I think that has been a more difficult thing to manage than actually doing the research itself. It's easier to get the funding if you are going to do some new research, but to just support an existing system, I don't know how the university manages that.

CARGO: Would you say that Ralph has had a high degree of success with NSF?

WAMPLER: Oh, I think extremely high, yes. I think he has done extremely well over the years from that. He is an extremely talented individual and is certainly well recognized for that.

CARGO: When you have been up here at what you have been referring as primarily a teaching college, if you will, have you been teaching Icon in classes here?

WAMPLER: As soon as I got one running. We have a class called Non-numeric Applications with Lenny Mullens, who was here first. And since he was down at the U of A under Ralph earlier under SNOBOL, they had a SNOBOL course - a course on non-numeric applications, and SNOBOL was being used. Essentially I took that course over and started using Icon as the vehicle and have taught that ever since. I guess that's been seven years of teaching Icon up here. That's the only course I teach it in, although the students have used it in other courses for other projects once they have learned it. There's not enough students up here, particularly at the undergraduate level, to do a course on how Icon works internally or a course on extending Icon in some ways. But as a language to use, it has been very successful.

CARGO: Can you tell me more about the organization of the course?

WAMPLER: The way it's organized now is, since the students have typically come out of a background where they have had primarily C, some Pascal, and FORTRAN, they are not really aware of how much a language can really do for you. And so, the first five or six weeks of the course is an attempt to change their mindset and to get them to realize that there is really no need to write all these little routines, that the language does it for them, and also to change their way of thinking away from a Boolean variable oriented true/false environment to a success/failure type environment such as Icon. Once we have spent five or six weeks on that then we start to look at using Icon in various ways - the string processing, the list processing, the associative memory aspects of tables, some work on text processing, a lot of things with graphs (which Icon happens to work really well with) and graph processing - that type of thing. It's a junior-level, undergraduate course so it doesn't get heavily into the more sophisticated topics. But the students enjoy it and they come out with a pretty good appreciation of how a language can influence how you design things.

CARGO: Is that a quarter class or a semester class?

WAMPLER: It's a three-hour semester class - 16 weeks in there. There's a lot of hands-on experience - courses oriented around having a lot of homework, programs to write, where each week you may have several programs, four or more, typically, to write. And then they learn real quickly how nice it is to have a language where what was 50 lines in C is four or five lines, and so five programs doesn't seem quite so bad after the first two assignments. When they first come in they panic when they're told, oh, the first five programs to write on the first assignment; it scares them. But then they learn. I am happy with the course; I am happy with the language and the fact that I get regularly statements from people who have left here to go someplace that are saying, "I wish I had Icon to do my work at, to use for my work," and that type of thing. I think it's working in terms of what I want them to see in a language.

CARGO: Do they have a class project, either individual or group as part of that?

WAMPLER: I have done it several ways. I had them do a class project before; I had them do a LISP interpreter once - usually individually, occasionally in pairs, although the last couple times I have taught it I have not done that for a variety of reasons. One reason is I wanted to try and cover more material at more breadth and a little less depth. Next time I may do more depth now that the new Icon book is out, and go back to having a project and see how they do with that. This is not really related to Icon, but something we have discovered here is that our students are getting killed a little bit with too many projects. As undergraduates, if they have four or five projects in a semester they are dead. We all like to give projects because we think they are a really good way to learn, but if we all give them the students don't get them done. They tend to do one and blow off the others, or not do as good as they should and not get as much out of it. So that's the reason I have not done the projects recently.

CARGO: Prevention of cruelty to students.

WAMPLER: Yes, as one of the older faculty members here now, I am beginning to start feeling guilty. When I was first here I didn't mind at all working them to death, but I am not sure they are learning as well as they would if they could relax a little bit more. So I am not doing quite as much on projects with them.

CARGO: Do you have any motivation to make sure that when your students leave here they can keep their hands on Icon implementations?

WAMPLER: I try very hard to let them know that Icon is free. If they have PCs, I have the same stuff that is on the bulletin board up here, so I can give it to them a little more conveniently than having to go to the bulletin board, or put it on floppies for them. And I try to make them aware that it's available for them. Usually where they can't use it is in a situation where some manager has said, "You will not..." or more specifically, "You will use this language for your work, and you won't use anything else." But, yes, they're usually very aware that it is out there. Most of them, if they have their own machine, will have it on their home machine. That's certainly one of the real strengths of Icon: it's a very high level language that's readily available on a lot of platforms now. The students really enjoy having a fairly sophisticated language on their IBM PCs. We don't see very many people with Macintosh versions because it's just too expensive. If you want the standard one you have to get MPW, which is $100, or $150, and if you want

ProIcon then you have to pay for that. And it's more than most of the students are willing to pay for it. But the PC version, being free, is just snapped up real quickly.

CARGO: What versions does NAU have on their machines here. I should say, maybe turn that around, what machines does NAU have that are running Icon?

WAMPLER: Well, it's almost all in computer science. The Computer Center won't touch it. It's the same sort of thing with a manager saying, "It doesn't come with a warranty," or "It's public domain, and there's nobody to blame if something goes wrong so we're not going to have it on our machines." So the Computer Center won't go near it. We have it on almost all the machines we have here. We finally got rid of our own PDP. We are running MicroVaxen, DEC stations. Silicon Graphics Personal IRIS, it's on all of those machines. We just got some SPARC stations in. I will bring it up on those machines after this workshop is over for people to use. It's available on a Macintosh II and on the IBM PCs. The one you're sitting next to has it on it, for example, somewhere. So it's available essentially on all the machines that the college has here. I make sure of that.

CARGO: When you say the Computer Center won't touch it, do you mean that they won't put in on their machines, or they will prevent it from being on their machines?

WAMPLER: They will not allow it on their machines. Essentially, the Computer Center here is under the philosophy that, "It's got to come from a company that you can blame or call up for support, or we won't support it ourselves. It will just simply confuse people and will get in the way." I have used Icon to do a course scheduler for the students so they can enter what courses they want and when they don't want to take courses, and what sections they want to avoid, and it will show them all the schedules. It's popular on campus, but the Computer Center will not run it, because that would mean that they would have to bring up Icon, which is a language that doesn't come with VMS or whatever they are running, so they don't want it, which is annoying, from my point of view. So that we support now this course scheduler out of our department. Social and behavioral sciences uses it a lot in some other areas, and we are not really geared up to providing that sort of service to the campus. I am also about to do a graduation degree checker, which would be of great help on campus, but I am going to write it in Icon, so it won't be on the mainframes

on campus for a while, although maybe there will be enough interest that we finally get that policy changed.

CARGO: What are the mainframes that the Computer Center is controlling that they won't put Icon on?

WAMPLER: They have a large IBM - I have forgotten the model number anymore - and then a VAX 6400. The campus here as a whole has just now discovered workstations. Outside of the computer science here there's probably five or six on campus. And so it's still a mainframe-oriented approach, and the VMS is the main academic machine - the VAX; but that's restricted.

CARGO: Do you think if there were some support from DEC they would embrace it?

WAMPLER: Yes, oh, absolutely. I think if it came out on a DEC tape from DEC it would be put on the machine. But it's just that the Computer Center director has this philosophy, or approach to things, that hurts languages like Icon, which are actually superior products, you know - very well-supported in many ways but don't have a company behind them to go to. Who know, maybe ProIcon or Cat's Paw can come up with a VMS version and then they can provide it for the Computer Center.

CARGO: Do you know if the Computer Center uses GNU Emacs?

WAMPLER: They do not. There is one person... I don't want to indicate there's only one, but there is a person over there who likes Icon a lot - has it on his home machine, for example - likes GNU Emacs, has it up and running and so on, but he's essentially not allowed to export that to other users; it sits. No, they run EDT, or EVE, or one of the standard VMS editors. They have VI; they don't use that very much because, again, it's not what DEC says to use on the VMS, and that sort of thing. It's an old-style computer center environment.

CARGO: Do you think that typifies some of the typical resistance that some places have to using Icon?

WAMPLER: Yes. Oh, I think absolutely, particularly in industry. I think there are a lot of managers or people who

will go, "If there's not a company behind this, how do I know it's any good? How do I know that those people of University of Arizona are going to go, 'Ah, I don't care about it anymore,' and leave the project and there will be no support for it. How do we know if it's verifiable? Who do we go to if we don't like something?" Even though most of that's in place, it's well established, and there's certainly a lot of help for it. It's just like people buy IBM because it says IBM, or they buy DEC because it says DEC. They'll buy a product that costs them twice as much as something else because it has the right brand name. And I think that hurts Icon and other languages as well that come out of a school environment. It's almost like you have to suddenly form a company and start marketing that way in order to validate the whole concept in some people's eyes. Yes, I think it's a big blow; I think it's really hard to overcome that - certainly from just the comments I get from my students who are saying, "Well, I wish I could use Icon here but it's not allowed," or so on indicates that there would be even more use of it if some policies were changed.

CARGO: Going back to the question of Icon education again, have you seen much adoption of Icon in teaching environments - academic environments - either in non-numeric applications courses or comparison of programming language courses?

WAMPLER: I think if you see it you see it in comparative programming language courses. Most schools tend to have a dominant language that most of the work is done in. It's either LISP if they're an AI school, or a language like C if they're a systems development school, or ADA if they're a software engineering school. And so most of the courses end up being taught with an emphasis toward those types of languages. So where you would see it is in comparative programming language courses. What hurts Icon again is it's not showing up in programming language course texts very much yet. I think I have only seen it in one. And so it is difficult for an instructor, unless they are familiar with the language, to bring it in from outside, into a comparative programming language course. It's not in the textbook, so there's nothing for the students to see. But yes, I think that's where you see it most often.

You also see it in places where you have an instructor or professor who has used Icon and is familiar with it; then it becomes easier to teach. It's not a mysterious language off in the corner anymore that way. I don't know what schools use it really heavily at all. I don't even know that it's in our comparative programming language course right now, because it's not in a textbook. I have used it when I have taught the course in the past, but the current person

teaching it, I don't know if he is using it or not.

CARGO: Have you had much communication with other people who have taught Icon?

WAMPLER: No, I haven't. I suspect Ralph has a great deal. I am not one of the big players, and so unless people know that I am up here they're not likely to go, "Oh, I have a question. I will send something to Steve about that." And I tend not to get involved too much on the network on things. No, I don't hear very much at all. I would love to. Since I love to teach (I view myself as a teacher before other things), that's something I am interested in is using Icon. I have this secret desire to teach it in the data structures class and show people what you can do with data structures, the languages you would really set up for it. But that is not going to happen any time soon. But it would be nice.

CARGO: You had mentioned before about your semester class not being able to get into what you considered to be the advanced features of Icon. Could you summarize what you think those advanced features are that you aren't able to teach?

WAMPLER: Well, I cover all of Icon, all the way down to co-expressions and the whole nine yards. I think what I probably meant to say was not so much the advanced features but sophisticated uses of Icon, where you are really hitting it hard, rather than just using it as a language that's easier to use than C or Pascal, using it as a language because you have co-expressions and because you have goal-directed evaluation. I don't get into that as much as I would like. You know, we do some backtracking, but we don't heavily spend a lot of time on backtracking problems. And I don't spend a lot of time on really sophisticated string scanning - writing your own text editor or writing your own text format are topics that usually I don't get into in the class. But I do cover all of Icon. We cover the entire book, or the entire old book. I am looking forward to using the new book, because it introduces in my view the nicer features of Icon earlier. And so I think it's going to be easier for the students to change their mindset because they are going to see new, nice ways of doing things very early on in the text, and that may allow me to get into some things a little deeper by the end of the course.

CARGO: What do you think those nicer features are?

WAMPLER: I think the whole concept of programming around goal-directed evaluation, the ability to do your own suspending on procedures and resume... come back into an existing environment easily and just pick up where you left off - that type of approach to programming, rather than saying, "Well, every time I call this function I have to pass in what I want, you know, and tell it I want the third word now, or the fifth word now. I'd rather have it produce the words in some order." That type of feature. Some work on co-expressions, although they are a complex enough topic that I am not sure juniors or undergraduate students are really at the level where they make a lot of sense yet. Students like to subscript things. They have learned arrays, and they learned everything in terms of arrays, and so it's really hard for them to get to use "find" and "move" and "tab" and "upto" and all this. They seem to be much happier using "&subject," "sub I" to deal with it. And that's not an appropriate way to program in Icon. It's much more appropriate to think in terms of movement through a string - not so much oriented toward position and explicitly given position everywhere. High-level string scanning is certainly something I will be emphasizing now. It shows up much earlier in the book now. The students in the old book would learn "&pos" and "&subject" and subscripting of strings before they really learned all this nice facility for moving around in them. Now they will see that nice facility first in the new book. They won't be imprinted with, "Oh, I can always go back to my array subscripting," which is the first thing they will do on a test when you ask them a question that involves string scanning. What you will see as the answer is a little loop subscripting through a string a character at a time looking for the first character. And if they find that then they will go into another little loop looking for the rest of the string. The good student, of course, won't, but the typical student will want to fall back on what they have already had two years of when they get in a pressure situation, and so I would like them to leap forward when they are in a pressure situation instead. I am hopeful that I will be able to do a little bit better job of that.

CARGO: Do you see a typical set of conceptual problems, misunderstandings that people have when they are learning Icon that you have to overcome?

WAMPLER: Yes, one is, because it looks like Pascal and like C, it works like Pascal and C. They always tend to do that. They don't think in terms of every loops; they think in terms of "while" loops with iterators, because the "every"

doesn't feel quite natural to them. If they do have an "every" loop they will always have a subscripted variable - every I = 1 to 10, even though they may not need to count. They may just be able to generate their results directly in 1/3 the space. And so the biggest conceptual hurdles are the fact that they have got already a mindset that comes out of traditional languages, and they'll map their understanding of Icon back into that set and then use Icon the same way. I have had some very good students - students I have had in other courses I know are very good who have really not done what I consider well in Icon. They have programmed just fine; their things run. But if you look at their program you are looking at Pascal. They are really strong Pascal programmers, and every loop will have to have a subscript, a subscripted variable just like a "for" in Pascal does. So the one big area is that.

The other one is just general, goal-directed evaluation - the fact that, "It's all right if this fails," and if it fails you can just pick up with an alternative rather than explicitly saying, "If this does not work then do this. Otherwise do this." Or that type of construct - the fact that you can say, "Well, do this or this." Or, "Do this and this," type of thing, and viewing it as a non-Boolean construct.

The array processing that they have had in the past influences their use of tables. They don't necessarily see how easy it is to set the table, to keep track of some data. Instead they will map string information into integers that they can then use as subscripts, instead of directly saying, "Well, gee, the string works just fine as a subscript," and then going on. Those are the big hurdles.

Co-expressions tend to just lose them. It's very hard for them to see how co-expression provides a new way of thinking about a problem for them. You see, my philosophy is that one of the things Icon gives you is a new way of thinking about things, and it's fun to explore the new way. Most undergraduate students are more concerned with, "I need to get out. I need to graduate," rather than thinking about a new way of doing things. So it's hard to get them sometimes to realize it's fun just to look at something different and to turn your mind around and look at it in a different direction. Sometimes you will look at it and say, "I like the old way better." Other times you will see a new way. Very often you will go back and you will write better C code once you have learned Icon, because you have just simply changed how you think about problems. Instead of worrying about all the little details that you need to do in C you see the general flow, and then you fill in the details. The mail I was responding to earlier today was from

a person who had a C program that was very complex, and it was easy for me to see a much simpler way to do it, and I attribute that a lot to looking at languages like Icon, where there are other approaches than saying, "Well, do I have to do it this way?" you know, or, "Isn't there another way to view the problem that produces it?" And that's the whole goal of the, really, non-numeric applications courses - to get students thinking that way. You can tell I like to teach.

CARGO: Yes.

WAMPLER: On that topic I will go on.

CARGO: I think fundamental to making Icon more popular is making it easy to teach. Do you think there is something that is a major stumbling block that could be overcome by making Icon easier to teach, somehow?

WAMPLER: Not really. To me, the only really conceptually tricky thing is co-expressions, and then it's primarily because the sorts of things co-expressions are naturally could be used for are complex problems. It's not so much the language; it's the fact that you're dealing with a problem itself that's hard. It might not be any easier to come up with a solution in a language that didn't have co-expressions in it. In fact, in my mind it would be harder, but it still is a hard concept to get across - particularly people who are used to sequential processing. No, I don't think so. I think the biggest thing is the fact that it's just not discovered. It's an easy language to learn. The books that Ralph and Madge have written are very easy to read; they're easy to learn from. The new book I am very happy with. I am very anxious to see how the students take to that. I think it's just simply not a standard language. "It's not ADA; it's not C; it's not Pascal; what good is it?" I mean, that sort of attitude is what you have to overcome. "Why do I need to learn another language? I can do everything I want in FORTRAN" - that sort of thing is what's hurting a language like Icon. You can convince people they need to learn C in addition to FORTRAN, because there's a lot of C in use out there. You can convince people to use ADA, because there's going to be a lot of use of ADA. But it's hard to convince somebody to use a language just because it's better for the job. They go, "Oh, no, I am going to have to program in C when I graduate. I don't want to take a course that doesn't help me in my job when I graduate."

CARGO: Do you think there is something that can overcome that?

WAMPLER: Time and - you know, I don't know what really causes a language to take off - a more efficient implementation so that managers start looking at it. What you will see is that they will go, "C is so much faster," or whatever is so much faster than Icon, even though for the type of work you are doing it may be much faster to develop in Icon this one-shot program. They measure things. It's like a test to measure machines. It's a number you can come up with, and if you can measure a language in terms of how fast it runs, then you can say whether it's good or bad, easily. And I think the Icon compiler will actually have an impact because of that, because it will pull Icon down into a closer range with a non-interpretive language. And that may change some people's minds about how it's perceived.

CARGO: Does the name Tim Budd ring any bell?

WAMPLER: Oh, yes. Yes, Tim and his language CG, which was way back, where he took generators and essentially put them into C. He was a very prolific person; he did a lot of work in other languages as well. And he showed that, really, things did not run particularly much slower than C; that is, the idea of a generator is not necessarily an inherently inefficient idea. I mean, CG was a fast language, but you were still programming in C. It wasn't enough of a change to really stand out as, "Here's what Icon can do for you," type thing. Yes, he was involved purely out of his own interest in Icon and playing with languages. He has written Little Small Talk, APL compilers, CG - looking at different languages that way. He was actually John Placer's advisor when John came down here and developed his own language based upon Icon.

CARGO: I was wondering if there might have been a relationship there. Was Tim associated with the University of Arizona at all?

WAMPLER: Tim was there for the last year or year and a half I was there. I never had him in a class. I knew him informally as a professor who was there and on his work with CG. He was never directly involved in the Icon Project, on Icon other than the fact that he just liked to play with programming languages a lot.

CARGO: Anything else that you can think of?

WAMPLER: No, I am worn out.

CARGO: Okay. I guess that will be it then.

END OF INTERVIEW