An Interview with

KENNETH WALKER

OH 205

Conducted by David S. Cargo

on

26 July 1990

Flagstaff, AZ

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

Kenneth Walker Interview
26 July 1990

Abstract

Walker describes the work environment of the Icon project. He discusses the process of language development, his interactions with Ralph Griswold, the level of staffing at the project, and his own work on Icon development, especially on an Icon compiler.

KENNETH WALKER INTERVIEW

DATE: 26 July 1990 INTERVIEWER: David S. Cargo

LOCATION: Flagstaff, AZ

CARGO: Ken, to start what I want to do is get some information about your background before you got involved with the Icon Project and then start with how you got involved with the Icon Project, and eventually what you did for it.

WALKER: Okay. How far back do you want to go?

CARGO: College.

WALKER: College, okay. I went to Carnegie-Mellon in 1973. I took my first programming course in I think it was the spring of 1974. Basically, I was majoring in mathematics, but after taking my first programming course, I pretty much concentrated in computer science. After graduating, I went to work for the Vermont State government as a computer programmer. I spent three years as an applications programmer in mostly FORTRAN and COBOL. Then I became a systems programmer working with the CICS telecommunications system as basically doing systems, sys-gens, and problem determination and so forth. I did some assembly language programming there and some programming in the CICS Exec language. Then after three and a half years there I decided to go back to graduate school and chose the University of Arizona because they seemed to be doing a lot of work in programming languages and operating systems, which were things that I was interested in. So I came here.

At the end of my first year, I was offered a research assistantship under the CER grant, so I talked with a variety of professors, or several of the professors on things they were working on to see who was doing things that I was also interested in. Ralph was working in programming languages and that's something I was interested in. And even though I didn't know Icon at the time he was willing to take a risk that I could learn it quickly, so I took the Icon book and in about two or three weeks I pretty much had the basics of the language down. I had already worked a little with

Prolog so goal-directed evaluation was something I was familiar with.

At that point, the first project I started working on was Cinema, which is a cinematic display of string scanning. Ralph had already done one version that used a regular character-oriented terminal, and basically it showed the program text and somehow indicated where string scanning was and what the subject string and the current position were. It showed the subject string in current position, and I don't remember all the details of what his particular version at that time looked like. John Placer had started working on a revised version, but just about the time I came on the project he took a job and left. So I took over that. Ralph and I decided that we would go to a Sun workstation because it gave us a better display and we could do at least some simple graphics. While working on that I developed a model for keeping track of both nested string scanning and resumption failure and resumption of goal-directed evaluation in terms of leaving and entering scanning environments, which turns out to be sort of tree-like thing where things can be suspended and string scanning can be nested.

Let's see; what else was there? Oh, there was always this problem in Icon of trying to describe, or trying to implement, sting-scanning in Icon itself. In the implementation there was this B-scan operator, or E-Scan virtual machine structure, and E-Scan, which begins scanning, and ends scanning, and they sort of have to communicate in terms of saving and restoring scanning environments, where scanning environments is a subject and position. And one day with a meeting with Ralph we were going over how to do this. It had been done with co-expressions, which was really awkward, and at that point I had this insight that by nesting expressions in a certain way, nesting procedure calls, you could actually communicate. You could write B-Scan and E-Scan in Icon and pass a scanning environment from one to the other, so you could essentially raise string scanning up to the Icon level, and that helped capture the events, made it much easier to display. So that was about all that there was in that project.

My next thing I worked on was an attempt to generalize string scanning. The particular approach I took was, okay, you have these scanning environments consisting of a subject and a position. Well, why don't I generalize them and have an environment with any of the variables you want, which aren't necessarily subjects and positions. And so you could declare an environment. What I came up with was sort of a special case of an object - in an object-oriented

program. And you could specify actions to take when execution entered one of these environments, and actions to take when execution left. And it was automatically this state [?] maintenance. String scanning acts very much like Dynamic Scope, because when you enter string scanning it sets up a subject... a recent [?] subject and a position... a recent [?] pause. And it does so dynamically and if you call both the functions or procedures, they see this new state. So it looks very much like dynamic scope, and it's the only place in Icon where you have this kind of dynamic scoping. So the scanning environments, or the generalized scanning environments, or generalized environments had this built-in feature where the fields of the environment were dynamically made visible.

Let's see, after that, as I recall, the type inference pretty much came next. So was there something else in between? One summer we were talking about efficient implementation, and I had just taken a compiler course and learned about global data-flow analysis, and a little tiny bit about type inference, and we were discussing how in the Icon implementation every operation type-checks all its arguments, even if they're constants, which, even though it was sort of hard to know how much execution time that was taking up, it certainly seemed like it would be a good idea if you could eliminate it. So I went off and started studying global data flow analysis more than I had and started coming up with a type inferencing system. And originally I had a rather ad-hoc type inferencing system, which I did implement. I prototyped it in Icon, which turned out to be a good language to prototype it in. And at about that point I guess I learned about abstract interpretation. So about the time I was winding it up I learned about the abstract interpretation as a way of viewing data flow analysis, or type inference, in particular, it's good at. So I then cast the type inferencing system in terms of global data flow, or in terms of abstract interpretation, which gave me some more insights and I went back and modified it.

During this period of time... the time frames are sort of hard to remember how things come together. I did a fair amount of work on implementing Version 7. And there was a lot of new features, so I was working on that at that time. There was a bug in string scanning, which in the first edition of the Icon book had actually been documented more as a feature than as a bug. It's where if you leave string scanning prematurely through a break or return, the subject and position where not reset to the outer environment. And based on what I had learned during Cinema and simulating string-scanning, I used that insight to figure out how to solve that particular problem. So that's now no

longer a feature; it is now a bug in a previous version.

The point of the first Icon workshop - I had the prototype type inferencing system written up. I had also decided that it was reasonable to use a register based model, or I should say temporary variable-based model, rather than a stacked-based model of execution for holding intermediate results, because when you suspend you have to copy the top portion of the stack, which seems very unstack-like. So I had investigated that. One of the problems there, which has been known from the beginning is that intermediate values in Icon have these extended lifetimes, because execution can backtrack to a point between where an operand is computed, and where the operand is used and continued forward again. So if you want fine-grained lifetime analysis you have to locate where backtracking can come from, sort of find the farthest backtracking point within a particular boundary expression, and then locate where things can generate, where execution can come forward again. So that was also pretty well done. I think I had a prototype of that working - not correctly; there were some bugs in it - about the time of the first workshop.

So at the first workshop I sort of promised to do this Icon compiler, and I decided that I wanted something at least production quality in terms of what the Icon Project thinks of production quality - something that is distributable; maybe not something that industry would consider as a product, but something distributable. And so I have spent the last two years programming that. It turned out to be a bigger project than I had anticipated. And part of the problem is that I had wanted to do something which would be easy to use and easy to add new built-in functions and features to it. So that consumed a fair amount of time and I was pretty fussy at the beginning in terms of testing things - maybe not quite as much now.

I guess that pretty much brings me up to date. At the moment, the compiler is generating code. It generates C-Code. It uses a C compiler as a back end. Type inference in place, and as of last Sunday I was starting to get optimizations out of type inference. The temporary variable based allocation is in with the lifetime analysis, and that seems to be working. Those sorts of things haven't been tested too much and there are more optimizations to come out of type inferencing before I start on my dissertation. I guess that's where I am.

CARGO: Just backtracking a bit, when did you graduate?

WALKER: Oh, from Carnegie-Mellon?

CARGO: Yes.

WALKER: In 1977.

CARGO: What year did you come to the U of A?

WALKER: 1984.

CARGO: And this is a bit more global question. For the stuff that you have been working on with Icon, how was it decided what it was you would be working on?

WALKER: Okay, for Cinema the project was sort of sitting there. John Placer was leaving, and it was something to sort of get my feet wet and get started on. The generalizing string scanning was a project that I decided sounded interesting. There was a period of time after Cinema was done when Ralph sort of let me go off on my own and try to think of something interesting to do. And so when I came up with that he felt that was interesting and set me to work with that. The type inferencing work was one of the summer sessions where there were at least four or five of us, in addition to Ralph. Let's see, who was definitely there? Ralph was there, Janalee O'Bagy, Kelvin Nilsen, Dave Gudeman. Gregg Townsend was probably there; he usually participates in the summer sessions. I think that's about it. And as I say, we were talking about ways of improving performance, improving the implementation. When the issue came up of trying to determine the types of expressions and eliminating type-checking, I volunteered to work on it, mostly because I had just come out of the compiler course and it sounded like really interesting stuff. I, of course, didn't anticipate how much work it was going to be, but... I have done routine maintenance on Icon. I have helped add features. Those are things that Ralph asked me to do. I also do things for the Icon Project outside of

programming and outside of research a little bit on the side. I answer certain routine e-mail. The secretaries usually transfer calls to me rather than bothering Ralph. Some of them are slightly technical; some of them are very technical. Some of them I can answer and sometimes I have to go out and ask someone else. So that takes up a little bit of my time. Any other questions to ask?

CARGO: When you are working on stuff for the Icon Project, does it tend to be things that you work on by yourself, or have you had team efforts or things like that?

WALKER: I have worked mostly by myself. Some of the implementing particular features, there has been coordination - especially with Version 7 when we had lots of new features coming in, there was some coordination from Ralph and me as to things going in. But mostly, I have worked on things by myself.

CARGO: So it sounds like Ralph does a lot of central coordination for however many people he has working on a current version.

WALKER: Yes, and with one central source that can be quite difficult if you have several people actually on the production, the distributed version, of Icon. At one point we did have sort of a cloned-off version - an experimental version. Which reminds me, there was one other project that I worked on. In one of these discussions that we were talking about the summer sessions, we were talking about features, and one of the things that Ralph has never been completely satisfied with is string scanning, because it's not as high level a feature as some of the SNOBOL4 patterns. There were several of us in the conversation. I am not sure who exactly instigated what ideas, but they came up with the idea of using co-expressions to essentially capture matching expressions, so that you could use them to dynamically build up a matching expression like you could build up a pattern. So I did work on that, and that involved a couple of changes. One, I changed the lifetime of local variables. In the distributed version of Icon, co-expressions always make a copy of local variables, so they have their own private copy of the variables local to the procedure that they came from. I changed it to be more like LISP closures where you have heap allocated local variables, and then several co-expressions can share them. This was only in the experimental version. And I also

augmented the Bang operation, so that if you did Bang on a co-expression (it would eventually be called C-expression to give it a better name), where activation produced one result at a time, Bang would essentially act like a procedure call on this co-expression and generate all of the results in one place. So it was actually being executed on the same stack as the Bang operation. It was just like you were using this piece of code, because that's more the semantics of pattern-matching or string-scanning; you want to generate all the results at all one time. So you are sort of throwing away some of the power of co-expressions and just using them to grab a piece of code. So that was sort of an aside.

And that came up, as I say, in the discussion, and I was interested and so I volunteered to do that. And that I did on my own, though there was a lot of group discussion on implementation and exactly what the semantics should be, and later on there were some discussions on extending this idea, though that never went any farther. This particular version of Icon was also the one that had the fully recursive interpreter, which was part of Janalee's dissertation, where currently when a built-in function suspends, it calls the interpreter recursively. This came about in Version 6 when we went to the Icon implementation, which was almost pure C, with just the context. And at that time there was also overflow checking on arithmetic done in assembler.

Unfortunately, the implementation book had already been revised once before being published and it was too late to revise it, although the recursive interpreter probably would have become the distributive interpreter, except that it would have been out of sync with the implementation book that was just being published, because the recursive interpreter is much nicer. Not only the interpreter get calls recursively when a built-in function suspends, but for other things like creating bounded expressions and built-in generators like alternation and repeated alternation. And it cleaned up the implementation, whereas before, when certain events occurred the interpreter had to go through this awkward thing of unwinding the stack and making sure the correct number of interpreters got unwound, whereas with the recursive interpreter model it was just very elegant. You just unwound, passing the signal back, and whoever the signal belonged to would stop the unwinding or the return sequence. And there were a few other things that were experimented with in that. It may still be around but it has sort of gotten lost. Nothing there to keep it mainstream.

CARGO: How did you folks go about your methods of configuration control? Seemingly, several people were working on the interpreter translator at the same time.

WALKER: At least my experience is that generally Ralph would say, "Okay, the central source is free. Go in and do your changes." That's been my experience, though of course there were outside people also making changes, and generally Ralph backed them in himself. For a while this cloned-off version was being updated and Dave Gudeman was doing like, was it three-way diff thing that allows you to merge features from a common ancestor (I forget exactly how that worked), which worked okay most of the time. But that was with the two separate interpreters. But my experience with the distributive version, Ralph would say, "Okay, it's set to go. Do it." He basically took care of coordination. Or if at a time Ralph had given me a big list of Version 7 features, I might have control of the source and I would have to give up control to someone else. So it was sort of like passing the baton kind of thing, where, "Okay, it belongs to you now."

CARGO: So you weren't using SCCS or CS on a UNIX system to keep control.

WALKER: No, it was all pretty much manual.

CARGO: I am curious as to what people you were working with when you started, perhaps in the middle, and which ones you are working with now, because it sounds like there has been considerable turnover.

WALKER: Oh, you mean, who was on the project?

CARGO: Yes.

WALKER: John Placer was just leaving as I came in. Janalee O'Bagy was working on the project when I came there. Kelvin Nilsen . . . I forget exactly whether he was on the project. It almost seems as though maybe he was, but I don't

really quite know how the coordination goes there. As I remember, the first summer we didn't have group meetings. It was separate meetings with Ralph. I don't think Kelvin was on the project. At some point Dave Gudeman was on the project for a while. Then Janalee O'Bagy and Kelvin graduated two years ago, just about the time of the first workshop. Dave Gudeman was on the project for a while after that. I think it was through the end of that summer. Then I guess I was pretty much on the project by myself. Clinton Jeffrey has been doing work off and on. He was doing some work I think that next summer, and then, as of this summer, Clint, Beth Weiss, and Nick Kline have been working full-time during the summer. Nina Bhatti has been here some of the summer. She has been attending meetings, and she did a literature search of, something to do with the graphical stuff. Of course, Gregg Townsend has been here, and he tends to participate in summer research meetings, and he also, as part of the lab staff, gets assignments to work on various things on the Icon Project. Bill Mitchell was here, once again, as a person on the lab staff, I believe, pretty much from when I arrived until he went to Sunquest, which was a year or two ago - maybe a year and a half. I forget exactly the dates. So I guess that's pretty much the people on the project that I have been working with.

CARGO: Do you find that Ralph's usual method of having people working on the project is to have them work as teams, or to work as individuals?

WALKER: I have seen mostly working as individuals, though this summer Clint, Nick, and Beth, and Ralph himself, have been working of different aspects of the visualization. And Gregg Townsend certainly has done active participation in the meetings, and I don't recall how much programming he is doing, because of course he is actually on the lab staff, and not a member of the Icon group, or the Icon research team anyway. There are certainly projects in which Ralph and another graduate student have worked together, each doing pieces of the work. On the compiler, when we went to recoding the run-time system, the run-time system of the compiler is essentially the same except that its coding is not pure C. It's sort of an extension of C so that the compiler system can recognize type-checking code and eliminate it, and that sort of thing.

TAPE 1/SIDE 2

WALKER: When we cloned the run-time system off to create the run-time system for the compiler, Ralph did some of the work to restructure the include files, which turned out to be somewhat complicated, because some of them had to come into different points in the translation. And Clinton Jeffrey did quite a bit of the work at actually translating from pure C to the sort of extended version of C last summer - that they got the majority of work out of the way while working half-time last summer. So there has been some of this coordination. As far as other things on the compiler, Ralph has done some of the work testing the compiler - earlier versions of the compiler, where up for ? very much optimizations, partly because he has this huge suite of Icon programs readily available to him, which I could get access to, but he knows where they are and how to run them. So he did that testing. I guess that's about how we coordinate things.

CARGO: Do you find that you use electronic mail a lot?

WALKER: I do.

CARGO: Why?

WALKER: It's quick and convenient, and if Ralph isn't available, you don't sit there waiting for him. So I communicate a lot. I find that to be a very way convenient way to communicate. But we usually have weekly meetings too. Well, at different points things have been set up differently. This summer I don't have regular weekly meetings. I do attend the Icon meetings, even though the only things discussed are the program visualization kinds of things. I meant, if I want a separate meeting with Ralph I have one, whereas back in the school year I had a weekly meeting scheduled with Ralph for an hour or so, or whatever. And most of the time I guess I have had weekly meetings, but there have been times when I have just sort of been off on my own for long periods of time, and if Ralph starts wondering what I am doing he just asks about it. That's mostly how we communicate. Sometimes, if we are not meeting regularly, I will try to come up with a weekly report or something... just the ?.

CARGO: Have you gotten involved with any of the efforts from people outside of the U of A doing things with the Icon translator or interpreter?

WALKER: I have given some advice, either through e-mail or over the phone, in some porting situations, in some of the ports. I know with the CDC machine I was carrying on both phone and e-mail discussions with the person who was working on that. That was a 64-bit port that was sort of going on at the same time the Cray port was being attempted. That's mostly, I guess, what I have been involved with. That's the main one that comes to mind. There probably have been some other things. The CDC one is the one I remember. I guess there was some discussions with the Cray, too. Most of the Icon ports to UNIX, particularly now, are really straightforward. If you have 32-bit integers with a byte-addressable machine, it usually goes pretty smoothly, but if you have strange addressing modes, or your addresses are a different size from integers, then things can get tricky. I think most of those problems are now solved. So, yes, I think that's about all of my interaction with the people outside.

CARGO: You mentioned early on that you had a grant when you started at the University of Arizona.

WALKER: I got the grant the first summer I was there without support. The original CER grant supported two RA positions. And Rick Schlichting, who, I guess, was in charge of finding the people for the grants, recommended me and suggested that I talk to faculty members to find someone to sponsor the research - be a research advisor.

CARGO: So that was a grant you got to enjoy. You didn't have to write it.

WALKER: Yes. The department decided that I was one of the people to get that, and I was, in fact, free to work with any professor who shared the interest and was willing to direct the research. Then eventually that ran out, and Ralph took over my funding.

CARGO: How many RA's have you seen simultaneously when you have been working with Ralph?

WALKER: You mean people on the project simultaneously?

CARGO: Yes, the research project.

WALKER: This summer it would be myself, Clinton Jeffrey, Beth Weiss, Nick Kline, and Nina Bhatti, I think, is half-time or something - quarter-time, maybe.

CARGO: What about when you first started working for Ralph off the CER grant?

WALKER: You mean how many people were on it at the time? I think there were only three of us. I know Janalee was on it and John Placer, myself. Let's see, at one time a little later there was myself, Janalee, Kelvin Nilsen, and Dave Gudeman. So there were four at one time at that point. As I say, at one point there was just me, so it has varied quite a bit.

CARGO: How long a period of time would you say it was just you?

WALKER: Maybe two semesters. I know Clint was on the project one summer. Well, yes, it would have to have been the project last summer. Now I don't really remember. I am pretty sure there was period of time when I was the only one on it. But it's not really clear in my mind - partly because Clint has been doing some work on Icon. He has also been working on the Emerald Project. It is not always clear to me when he is an RA and when he has just been working. In fact, he may have been on the Icon Project the first year he came here for a while. Yes, it's really hazy in my mind.

CARGO: I was wondering if there was any particular language or interpreter features that you had the idea for and wound up getting them included in one of the versions of the system.

WALKER: Mostly, I think I have just done experimental features that never got in the mainstream language.

Nothing comes to mind. I feel my major contribution that comes to mind is the fixing of string scanning, with scanning environments. But as far as real features, nothing comes to mind. I certainly helped move quite a few of them from Version 7.

CARGO: What kind of specifications were you given to implement, or to describe a feature to implement? How did you know what it was you were supposed to implement?

WALKER: Well, Ralph has what he calls his job jar, job jars, or whatever. And included in that is people's wish lists, so basically there's just a short verbal description. Sometimes we discussed features. I remember there was a discussion of variable length argument, or various procedures with variable length arguments. Let's see, how did that one go? That was I think during one of the summer sessions. There was a discussion of what the future should look like. And then two people - I think it was Dave Gudeman and Bill Mitchell, if I remember right, went off and in a few minutes hacked up a version of it, which eventually became in the language. I think they had their own clone version. Eventually I backed it in and made some changes. So I guess that one was pretty much already decided on. Basically, the written description of what somebody wanted and then either Ralph and I, or a group of us, would discuss what the feature should really be like. I mean, some of the things were maybe obvious, and some of them were maybe less obvious. It's actually hard to remember back to who exactly did which features or who did what work. I hear we were doing a lot of programming. [laugh]

CARGO: Once you knew what feature you were trying to implement, how was implementation strategy decided, what kind of changes to make?

WALKER: If it was like cloning techniques or whatever, I pretty much decided myself. If I felt there was a question as to what Ralph would want done, then I would send e-mail to Ralph, or talked to him at a meeting, or whatever. Certainly if it was a matter of the semantics of the operation, then I would definitely send mail to Ralph or talk to Ralph about it. If it was impact on the implementation, if I thought it was a significant impact, or a significant change in the way things were done, then I would generally consult with Ralph. I would generally put forth the proposals, or

maybe some alternatives and Ralph would say yes to this, or whatever. I don't remember too many situations where Ralph didn't like what I had done after I had done it. I tried to be sort of careful about that.

CARGO: When you were actually down coding, was there some aspect of that coding that was foremost in your mind when you were doing it? Making sure that it was portable, or that it readable, or easy to change, or efficient?

WALKER: This isn't foremost. For me, readability is probably pretty important. Portability - that came up. Portability is something that we had a lot of struggle with doing... figuring out how to do pointer arithmetic right, and it was done wrong several times. It would work on a typical UNIX system with 32-bit ints, and 32-bit pointers, and character addressing and all of that. It was real problems going to MS-DOS, which I really wasn't involved much with, where you have to use long words. If you have 32-bit addresses, (well, for the large memory model anyway, which is what we were using at the time... well, it was actually both large and small) you needed to have integers and pointers be the same size. The problem is with the segment registers in MS-DOS, which is a real problem for everybody to be programming in MS-DOS. And there were at least, I think, a couple of attempts to try to do that right.

And at one point the addressing was set up so that it was working under MS-DOS, but there were systems like the Cray, and I think CDC too perhaps (I am not sure which machine that was) where it didn't work. So there was a set of guidelines that I was trying to follow, which turned out to be adequate for MS-DOS, but not ? ? . That was certainly always going on. The Icon implementation tends to be pretty well written, and I certainly tried to keep the code at least as readable as it already was. And there are certain coding conventions in the Icon Project, which at the point we are doing Version 7, I guess there wasn't anything written down, but there were certain things like always indenting three characters, and don't indent using tabs because it messes up something. I think maybe that was the program for automatically extracting source for examples of the implementation, or something. There's a bunch of tools that go and look at the source like that, so there's certain conventions that are followed there. So I tried to follow them.

As far as efficiency goes, that was pretty important too. I guess all the things you mentioned were important, and I don't know that one was more important. I probably didn't think as much about, or probably even now don't think as much about, portability as maybe I should. That is a hard one to remember to keep all the things in your mind as to what you need or keep the pitfalls in your mind so that you avoid them. Any other questions?

CARGO: I think I am just about done, but is there a particular high point in your involvement with the Icon Project or the Icon Research Group that sticks out for you?

WALKER: I know when I first started getting executable code out of my compiler, even if it was only very simple expression, that was a pretty high point. I had been working a year on it. The Icon Workshop two years ago was a pretty high point, too; that was exciting. Those are the ones that come to mind.

CARGO: Is there anything about your involvement with the Icon Project that I didn't ask about that you want to make sure gets remembered?

WALKER: There's nothing that I can think of right now, anyway. That's about it.

CARGO: All right, then.

END OF INTERVIEW