An Interview with

Edsger W. Dijkstra

OH 330

Conducted by Philip L. Frana

on

August 2, 2001

Austin, TX

Edsger W. Dijkstra Interview

2001

Abstract

In this oral history Edsger Dijkstra recounts his early education and training as a theoretical physicist and as a ‘programmer’. Dijkstra describes his work developing software for the Mathematical Centre in Amsterdam, the completion of his Ph.D. thesis, and his activities at several early information processing conferences. Dijkstra also discourses on the development of ALGOL 60 and the origins of computing science in Europe and America.

Dijkstra: It all started in 1951, when my father enabled me to go to a programming course in Cambridge, England. It was a frightening experience: it was the first time that I left the Netherlands, the first time I ever had to understand people speaking English and immediately I was all by myself, trying to follow a course on a totally new topic. But I liked it very much. The Netherlands was such a small country that Aad van Wijngaarden, who was the director of the Computation Department of the Mathematical Centre in Amsterdam, knew of this, and he offered me a job, which I could only accept in March of 1952. And on a part-time basis, I became the programmer of the Mathematical Centre. They didn't have computers yet; they were trying to build them. I programmed until 1962 and the first eight years of my programming there I developed the basic software for a series of successive machines being built at the Mathematical Centre. And in those beginning years I was a very conservative programmer. The way in which programs were written down, the form of the instruction code on paper and the library organization, input, output, etc., it was very much modeled after what I had seen in '51 in Cambridge, but it's all worked satisfactorily.

Frana: It is my understanding that when you got married, you could not enter the term "programmer" into your marriage record?

Dijkstra: That's true. Yes.

Frana: When could that be recognized as a title?

Dijkstra: I think in the early sixties. I mean it's a well-known and true story. I was supposed to study theoretical physics, I did so and that was the reason for going to Cambridge, because I was being groomed to become an excellent theoretical physicist. The idea was that if besides all the usual tools I could also use computers, I could perhaps become a better theoretical physicist, be more than average. However, in '55 after three years of programming, while I was still a student, I concluded that the intellectual challenge of programming was greater than the intellectual challenge of theoretical physics, and as a result I chose programming, the reason being that programming was so unforgiving. If something went wrong, I mean a zero is a zero and a one is a one. I had never used someone else's software. I used machines that I had myself been involved in the design of, I had documented the instruction code and I had written my own basic software and all the programs and if something went wrong I had done it. And it was that unforgiving-ness that challenged me, because I also began to realize that though in principle it's very simple. A zero is a zero and a one is a one, and you should have complete intellectual control of what you are doing, in some strange and not understood way, programs could become very complicated, tricky. So it was in '55 when I decided not to become a physicist, to become a programmer instead. It was a difficult decision because, as I said, I was groomed to become a first-class scientist, and at the time programming didn't look like doing science, it was just a mixture of being ingenious and being accurate. I remember that I envied my hardware friends, because if you asked them what their professional competence consisted of, they could point out that they knew everything about triodes, pentodes, the valves and other electronic gear. And there was nothing I could point to! Actually, well, I had made a number of programs and I had a very good handwriting…

Frana: So, how did you go about making computing science respectable then?

Dijkstra: Well, that happened later. I spoke with van Wijngaarden in ’55, and explained my dilemma that I had to take leave from science if I became a programmer. And he said, well, first of all he pointed out that computers were here to stay, that it was not a fad. And then he said he agreed that there was no such thing as a clear scientific component in computer programming, but that I might very well be one of the people called to make it a science. And at the time, I was the kind of guy to whom you could say such things and would even accept them. As I said, I was trained to become a scientist. When I came in 1952, they were working on the ARRA [= Automatische Relais Rekenmachine Amsterdam], but they could not get it reliable, and an updated version, using selenium diodes instead of relais, was built. And then the Mathematical Centre built a machine for Fokker Aircraft Industry, because before that we had done on the ARRA lots of calculations for the F27. So the FERTA [= Fokker Electronische Rekenmachine Te Amsterdam], an updated version of the ARRA, was built and installed at Schiphol. The installation I did together with the young Gerard Blaauw who later became one of the designers of the IBM 360, with Amdahl and Brooks. One funny story about the F27, which Fairchild later built under the name “Friendship.” On my first visit to Australia, I flew a big 747 from Amsterdam to Los Angeles, then I changed planes and I got on another 747, from America I flew to Sydney or Melbourne, I don’t remember which of the two. And then it was the final part of the journey on an F27 to Canberra, because I had been invited by the Australians to Canberra. And we arrived and I left the plane and I met my host, whom I had never met before, but we had corresponded extensively. And he was very apologetic that this world traveler had to do the last leg of the journey on such a shaky two-engine turboprop. And it gave me the dear opportunity for a one-upmanship that I never got again. I could honestly say, “Dr. Stanton, I felt quite safe: I calculated the resonance frequencies of the wings myself.” [laughter]

Frana: Now was this the same journey that you continued around the globe? You stopped in India?

Dijkstra: No, no, this was a different trip, that was the trip to Japan. Okay, in ’54 the completion of the ARRA, in ’55/’56 I become a programmer. Then, in 1956, all sorts of things happened. I got my degree as a theoretical physicist. As soon as I had decided to become a programmer I finished my studies as quickly as possible, since I no longer felt welcome at the University: the physicists would considered me as a deserter, and the mathematicians, they were dismissive and somewhat contemptuous about computing anyhow. It was all finite, wasn’t it? In the mathematical culture of those days in the Netherlands you had to deal with infinity to make your topic scientifically respectable. In 1956 I did two important things, I got my degree and we had the festive opening of the ARMAC [= Automatische Rekenmachine MAthematische Centrum]. We had to have a demonstration. Now the ARRA, a few years earlier, had been so unreliable, that the only safe demonstration we dared to give was the generation of random numbers, but for the more reliable ARMAC I could try something more ambitious. The problem of course is that for a demonstration for non-computing people you have to have a problem statement that non-mathematicians can understand, even they have to understand the answer. So I designed a program that would find the shortest route between two cities in the Netherlands, using a somewhat reduced roadmap of the Netherlands, on which I had selected, 64 cities (so that in the coding, 6 bits would suffice to identify a city).

Frana: Somewhat similar to the seven-bridges problem?

Dijkstra: No, it's just the shortest route. What's the shortest way to travel from Rotterdam to Groningen, in general: from given city to given city. It is the algorithm for the shortest path, which I designed in about twenty minutes. One morning I was shopping in Amsterdam with my young fiancée, and tired, we sat down on the café terrace to drink a cup of coffee and I was just thinking about whether I could do this, and I then designed the algorithm for the shortest path. As I said, it was a twenty-minute invention. In fact, it was published in '59, three years late. The publication is still readable, it is, in fact, quite nice. One of the reasons that it is so nice was that I designed it without pencil and paper. I learned later that one of the advantages of designing without pencil and paper is that you are almost forced to avoid all avoidable complexities. Eventually that algorithm became, to my great amazement, one of the cornerstones of my fame. I found it in the early '60's in a German book on management science - - "Das Dijkstra'sche Verfahren." Suddenly, there was a method named after me.

Frana: It's of economic value.

Dijkstra: Yes. And it jumped again recently because it is extensively used in all travel planners. If, these days, you want to go from here to there and you have a car with a GPS and a screen, it can give you the shortest way from your hotel to Robbie Creek Cove. Yes?

Frana: I did generate a map, yes, on line.

Dijkstra: Did you use it?

Frana: Yes, I did use it.

Dijkstra: Then you used my algorithm this morning.

Frana: When was that originally published?

Dijkstra: It was originally published in 1959 in a German article in *Numerische Mathematik* . [1] F.L. Bauer had founded that journal and he was soliciting for articles. Now, at the time, something like an algorithm for the shortest path, that was hardly

[1] E.W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik* 1 (1959): 269-271.

considered mathematics: there was a finite number of ways of going from A to B and obviously there is a shortest one, so what's all the fuss about? So that is why it remained unpublished, but when Bauer asked whether we could contribute something, I said, yes, I could, because in the mean time I had also designed the shortest subspanning tree. I had done that for my hardware friends. You know, on the big panel you have to connect a whole lot of points with the same copper wire because they have to have the same voltage. How do you minimize the amount of copper wire that connects these points? So I wrote an article "A note on two problems in connection with graphs" or something like that. But now, when I went to my ophthalmologist - - and to the best of my knowledge he did not even know that I was a computing scientist - - he said, 'Have you designed the algorithm for GPS?" It turned out he had seen the *Scientific American* [2] of last November.

You asked a question…

Frana: What did you do with your fame then?

Dijkstra: What did I do with my fame? I was amazed by it!

Okay that was '56. '57 was the year of my marriage, which I had to do as a professional theoretical physicist, crazy story. Besides that I was confronted with the real-time interrupt, and I panicked almost. Because, you see, for those first five years I had always been programming for non-existing machines, not yet existing machines. I would discuss, we would design the instruction code, I would check whether this was an instruction code I could live with, and my hardware friend would check that this was an instruction code that they could build. And then I would write down the formal specification of the machine, and all three of us would sign it with our blood, so to speak. And then our ways parted and they would start, they knew what to build and I knew what I could construct. But all the programming I did was on paper. So I was quite used to the fact of developing programs without testing them.

Frana: There's no way to test them?

Dijkstra: There was not a way to test them, so you've got to convince yourself of their correctness by reasoning about them. Now, of course, there could be writing errors, I never claimed infallibility. You have to leave that to experts in that field like the Pope. But a simple writing error did not matter as long as the machine wasn't there yet, and as soon as errors would show up on the machine, they would be simple to correct. However, in '57, I was confronted with the idea of the real-time interrupt and that created a vision of a program with non-reproducible errors, because a real-time interrupt occurs at an unpredictable moment. My hardware friends flattered me out of my resistance. They said, "Yes, yes, we see your problem, but surely you must be up to it…" I learned to cope with it. I wrote a real-time interrupt handler which was flawless and which became the topic of

[2] Michael Menduno, "Atlas Shrugged: When it Comes to Online Road Maps, Why You Can't (Always) Get There from Here," *Scientific American* 283 (November 2000): 20, 22.

my Ph.D. thesis.[3] Later I would learn that over here, this would almost be considered an un-American activity.

Frana: Why is that?

Dijkstra: Well, the American reaction was very, very different. When IBM had to develop the software for the 360, they built one or two machines especially equipped with a monitor. That is an extra piece of machinery that would exactly record when interrupts took place and from where to where. And if something went wrong, it could replay it again and use the recorded history to control when interrupts would occur. So they made it reproducible, yes, but at the expense of much more hardware than we could afford. Needless to say, they never got the OS/360 right.

Frana: That's true. That's very true.

Dijkstra: Because …

Frana: Now that would have never occurred to a European?

Dijkstra: No, we were too poor.

Frana: To consider it?

Dijkstra: We were too poor to consider it and we also decided that we should try to structure our designs in such a way that we could keep things under our intellectual control and that there was no need to search experimentally for bugs. This was a major difference between European and American attitudes about programming to which I will return later.

Okay, in 1958, I wrote the basic software and I wrote my thesis about it. I only had the oral defense a year later because I wrote my thesis in Dutch. Those were the days when I could not write fluently in English, and it was translated with the aid of a South African colleague. In 1959, I published those two algorithms, shortest path and the shortest subspanning tree. I had my oral defense and became Dr. Dijkstra. And there was one illuminating experience at the Mathematical Centre. I had challenged my colleagues with the following programming task. Consider two cyclic programs, and in each cycle occurs a section called the critical section. The two programs can communicate by single reads and single writes in a common store, and about the relative speeds of the programs nothing is known. Try to synchronize these programs in such a way that at any moment in time at most one of them is engaged in its critical section. It's an implementation of the mutual exclusion problem which later became a cornerstone of THE Multiprogramming System - - underlying the monitor concept and a whole lot of things. I looked at it and realized that it was not trivial at all, there were all sorts of side conditions. For instance, if

[3] E.W. Dijkstra, "Communication with an Automatic Computer," Ph.D. diss., University of Amsterdam, 1959.

one of the programs would stay for a very long time in its non-critical section, the other one should be able to go on unhampered, so going critical alternatingly was not allowed. Another thing that was not allowed was what we called 'After-you-after-you' blocking, where the programs would compete for access to the critical section and the dilemma would never be solved. Now, my friends at the Mathematical Centre handed in their solutions, but they were all wrong. For each purported solution I would sketch a scenario that would reveal the bug. Well, what happened then was that people made their programs more sophisticated, more complicated, and one or two days later they would come down with their next version. It was wrong as well, only the construction and counter-examples became even more time-consuming and I had to change the rules of the game. I said, “Sir, sorry but I can no longer spend all my days refuting your vain efforts. From now onwards I only accept a solution with an argument why it is correct.”

Frana: With the program correctness?

Dijkstra: Yes, and within three hours or so Th. J. Dekker came with a perfect solution and a proof of its correctness. He had analyzed what kind of proof would be needed. What are the things I have to show? How can I prove them? What would be an acceptable structure of the correctness proof? Having settled that, he wrote down a program that met the proof's requirement. Which very clearly shows that you lose a lot when you restrict the role of mathematics to program verification as opposed to program construction or derivation. That was what we learned from the competition that was won by Dekker.

(Excuse me for a moment.) . . . . .

Another experience of 1959 was the Zeroth IFIP Congress in Paris, a precursor under the auspices of the United Nations. I attended that, though I was still a not too well-known youngster. I had visited England a few times, I had visited Germany a few times. My international contacts had started in December '58, when the preparatory meetings for the design of ALGOL 60 started. Normally the Mathematical Centre would have been represented by my boss, Aad van Wijingaarden, but he had had a serious car accident, and J.A. Zonneveld and I, as his immediate underlings, had to replace him. Zonneveld was a numerical analyst and did the numerical work, while I, as a programmer, did the programming work. And so I began attending the preparatory meetings for ALGOL 60. That was about the first time that I had to carry out spontaneously discussions in English. It was tough.

Frana: How did you learn?

Dijkstra: Well, I had, of course, had English in school, in high school. I had read a lot, I had no experience in writing or speaking it. I learned an awful lot from friends as soon as they knew that I liked to be corrected. And I'm extremely grateful to Mike Woodger of the National Physical Laboratory, Teddington, whom I met in December '58 for the first time. He quickly made it a rule for himself that he would only correct those errors he heard me make the second time, because otherwise conversation would have been impossible. That greatly helped and I think that after three or four years or so, I was beginning to feel comfortable.

Frana: I heard you remark that learning many different languages is useful to programming.

Dijkstra: Oh yes, it’s useful. There is an enormous difference between one who is monolingual and someone who at least knows a second language well, because it makes you much more conscious about the phenomenon of language structure in general. You will discover that certain constructions in one language you just can’t translate. I was once asked what were the most vital assets of a competent programmer, and I said, at the time, a mathematical inclination and an exceptional mastery of his native tongue. I said “mathematical inclination” because at the time it was not clear how mathematics as understood at that moment could contribute to a programming challenge. And I said “native tongue” and not “English” because, well for lack of formalism, you have to think in terms of words and sentences using a language you are familiar with. Moreover, but I didn’t know that at the time, research has shown that when people acquire a second language, they never get a greater mastery of it than they have of their own language, their native tongue: that clearly sets the standard for how articulate you are going to be.

Frana: I know I’ve interrupted your chronology here. For frame of reference, how far away are we from the beginning of the 2.3 Working Group?

Dijkstra: Ten years.

Frana: Ten years later?

Dijkstra: Yes. That started in…

Frana: ’69 ?

Dijkstra: Yes. ’69, and we were at ’59 in Paris. One of the main things that I remember from that was that Alan Perlis of Carnegie Institute of Technology, I don’t think it was already CMU at that moment, tried to hire Peter Naur and me. He must have had good taste to pick us, to pick Peter and me out of that crowd of people. As soon as Peter heard what Perlis would like us to do, he became very contemptuous. He referred to the United States as “The Backward Country.”

Frana: Peter did?

Dijkstra: Yes. He said that if they wanted to embark on a silly project, they had better do it themselves.

Frana: I think he’s mellowed since then.

Dijkstra: I’ve known him for years.

Frana: He's a friend of Arthur Norberg's, the Director at CBI.

Dijkstra: Oh yes?

Frana: They are still in touch.

Dijkstra: They are? Now then, ALGOL 60 came, which in retrospect, I think, should be regarded as the beginning of computing science; if we wish to mark a discontinuity in the way in which we thought about computing, then that is the emergence of ALGOL 60. I will return to that later, to the extent in which it has made, for instance, the topic academically respectable.

Frana: Yes, I want to hear more about that.

Dijkstra: I had published a paper called "Recursive Programming," again in *Numerische Mathematik,* [4] which was crazy because it had nothing to do with numerical mathematics, but that was the medium then. In '61, I just started thinking about programming and was beginning to realize that programming really was an intellectual challenge. One of the things Peter and I did was to be main speakers at a workshop or a summer school that had been organized in Brighton, England. There was there some sort of study institute on automatic programming, organized by a certain Goodman; there were quite a number of well-known British scientists in that audience and I think that that has played a major role in me becoming known in Great Britain. In my audience I had Tony Hoare, and neither of us remembers that. I don't remember him because he was one of the many people in the audience, and he doesn't remember it because in his memory Peter Naur and I, both bearded and both with a Continental accent, have merged into one person. [laughter] We reconstructed later that we were both there.

Frana: You were both there?

Dijkstra: We were both there: he was in the audience, and I was one of the main speakers. Okay, now in 1962, that was when my thinking about program synchronization resulted in the P- & V- operations; that was one major step in 1962. The other thing I remember was a conference in Rome on symbol manipulation, in April or so. Peter was there, with his wife. And they, the girls, went to Rome during the day, and one day they came back and there was a reception at the end of the day; it was early and while we were walking through the crowd, they hear one guy summarizing to another the political situation: quite clearly, of course, there were the Americans, and there were the Germans, and there were the French and the English. "Et alors il'y a les deux Fidel Castros!" A hilarious moment…. There were panel discussions and Peter and I were sitting next to each other and we had all sorts of nasty comments, but we made it the rule that we would go downstairs to the microphone in turn - - when it was my turn, I would go and next time he. This had gone on for an hour or so, and van Wijngaarden, my boss, was sitting next to

[4] E.W. Dijkstra, "Recursive Programming," *Numerische Mathematik* 2 (1960): 312-318.

an American and at a given moment the American grabs his shoulder and says “My God! There are two of them.” [laughter] This may be included in an oral history? It’s not mathematics, it isn’t computer science either, but it is a true story…..

The next thing was that in September, I went to the first IFIP Congress, which was held in Munich, and gave an invited speech on advancing programming, and it was there that I got a number of curtain calls: clearly I was telling something unusual. I will return to that also later. Then I became a professor of Mathematics in Eindhoven, and for two years I lectured on numerical analysis. By 1963/64, I had designed with Carel S. Scholten the specification of the hardware channels and the interrupts of the Electrologica X8, the last machine my friends built, and then I started on the design of THE Multi-programming System, which was completed a few years later. Of course, ’64 was the year in which IBM announced the 360. I was extremely cross with Jerry Blaaw, with whom I had cooperated and who should have known better, because there were serious flaws built into the I/O organization of that machine.[5]

Frana: Which you had discussed with him before?

Dijkstra: No I hadn’t, but he should have known of my Ph.D. thesis, and he should have known about the care that has to go into the design of such things, but that was clearly not a part of the IBM culture. In my Turing Lecture I have described the week that I studied the specifications of the 360, it was [laughter] the darkest week in my professional life. In a NATO Conference on Software Engineering in 1969 in Rome, [6] I have characterized the Russian decision to build a bit-compatible copy of the IBM 360 as the greatest American victory in the Cold War.

Frana: Did that make the proceedings? I don’t remember.

Dijkstra: No. That did not make the proceedings, no; Brian Randell and John Buxton saw to that. But it got spread anyhow. I was given an announcement that for many years has been posted in a sovjet computation center. It was a Cossack dancing with a flag saying, “We want IBM.” and underneath you have that quotation, about the victory in the Cold War. And I had the pleasure of telling it to a Russian audience in Petersburg. Somebody asked me my opinion about the 360, and I thought this the most compact summary that existed and I gave it. And a third of the people laughed, so you knew how many of them understood English. My way of checking them. Okay now, 64-65. I had generalized Dekker’s solution for N processes and the last sentence of that one-page article is, “And this, the author believes, completes the proof.” According to Doug Ross, who wrote a review of the article, it was about the first publication of an algorithm that included its correctness proof. Whether that is right or not I don’t know. Yes, I wrote “Cooperating Sequential Processes,” and I invented the Problem of the Dining Quintuple, which later

[5] See E.W. Dijkstra, “Over de IBM 360,” EWD 255, n.d., circulated privately, url = “http://www.cs.utexas.edu/ users/EWD/ewd02xx/EWD255.PDF”

[6] P. Naur and B. Randell, editors, *Software Engineering: A Report on a Conference Sponsored by the NATO Science Committee* (NATO, 1969).

became known as the Problem of the Dining Philosophers.[7] That name was given to it by Tony Hoare. And I gave an invited talk at the IFIP Conference in New York.

Frana: That was your first trip?

Dijkstra: No, it was my second trip. My first trip had been in '63. That was to an ACM Conference in Princeton. And I later visited a number of Burroughs outfits, that was the first time I met Donald Knuth. Talk about fame - - in one way or another I must already have had some fame in 1963 at my first visit to the States, because that was an ACM workshop with about 60 to 80 participants and I was invited to join. And they paid me five hundred dollars. Five hundred dollars was quite a lot in those days, and there was nothing I needed to do: I didn't need to give a speech, I didn't need to sit in a panel discussion, they just would like me to be there. Quite an amazing experience.

Frana: Yes.

Dijkstra: That was the start of my visit. And then I found some other money to do some more traveling.

Frana: What do you remember of your first two trips to America that surprised you about the way the profession seemed to be operating?

Dijkstra: Well, the first lecture at that workshop was given by a guy from IBM. It was very algebraic and complicated. On the blackboard he wrote wall-to-wall formulae and I didn't understand a single word of it. But there were many people that joined the conversation, that joined the discussion and posed questions. And I couldn't understand those questions either, nor the answers, and, as I said, I had been invited to come and was even paid to come. I felt miserably out of place and in the evening, during a reception, I voiced my worry that I was there on false premises. "How come?" [I was asked.] "The first speaker, I did not understand a word of it." "Oh," he said, "none of us did. That was all nonsense and gibberish, but IBM is sponsoring this, so we had to give the first slot to an IBM speaker."

Frana: Very common.

Dijkstra: Well, that was totally new, totally new for me.

Frana: That a keynote speaker would be someone who really wasn't going to participate in the conversation?

Dijkstra: Yes, and everybody knowing this and everybody going through the motions. It was a degree of scientific dishonesty I was not prepared for. Or… "a degree of scientific

[7] See E.W. Dijkstra, "Hierarchical Ordering of Sequential Processes," *Acta Informatica* 1 (1971): 115-138.

dishonesty," that is a bit too strong of a term. Let's say that the fence between science and industry, the fence around a University Campus, is here not as high as I was used to.

Frana: Were public intellectuals and scientists under attack those first two trips, do you recall, the way they were a couple years later?

Dijkstra: Oh yes. It was on this trip, I think my first trip, that I learned the term "egghead."

The most hilarious memory has nothing to do with computing science. I was sitting in front of The Old Nassau Inn and there was a reunion of veterans of some war who wanted their picture taken and the wife of one of them would take the picture. However that was an old Polaroid Land camera, and you had to hold it some time and she couldn't do that: each time, a minute later, when the thing came out, it was all blurry, I looked and I had seen that. Then one of the guys hollers to me, "Hi, young man. Got a steady hand?" So this young man had a steady hand and took the picture.

In '66 was the completion of the design of the THE System, the Banker's Algorithm. '67 was the ACM Conference on Operating Systems principles in Gatlinburg. That, I think, was the first time that I had a large American audience. It was at that meeting, one afternoon I was just sitting outside, that I explained to Brian Randell and a few others one of the reasons why the GO TO statement introduced complexity. And they asked me to publish it. So I sent an article to the Communications of the ACM, called "A Case Against the GO TO Statement." The editor of the section wanted to publish it as quickly as possible, so he turned it from an article into a Letter to the Editor, which could be published immediately. And in doing so, he changed the title into, "The GO TO Considered Harmful."[8]

Frana: What was the original title?

Dijkstra: The original title was "A Case Against the GO TO Statement."

Frana: Well, that changes things considerably.

Dijkstra: Yes, and the title became a template. Hundreds of writers have "X considered harmful," with X anything. The editor who made this change was Niklaus Wirth.

Frana: Wow. That is a phrase that's entered …

Dijkstra: the lingo….

Frana: common lexicon … yes.

[8] E.W. Dijkstra, "Go To Statement Considered Harmful," *Communications of the ACM* 11 (March 1968): 147-148.

Dijkstra: Yes. Now to ’68. ’68 was exciting because of the first NATO Conference on Software Engineering, in Garmisch. In *BIT* I published a paper,[9] which remained unnoticed because the circulation of *BIT* was too small, on “A Constructive Approach to the Problem of Program Correctness.” Remember what I said about Dekker’s experience in ’59. ’68 was also the year of the IBM advertisement in *Datamation,* of a beaming Susy Mayer, in full color, who had just solved all her programming problems by switching to PL/I. Remember, those were the days we were forced to, led to believe that the problems of programming were the problems of the deficiencies of the programming language you were working with, and if you took the right language then there would be no problems anymore.

Frana: Iverson’s APL was unsatisfactory?

Dijkstra: Was it already there?

Frana: I believe so. Maybe it had not - - it got a slow start.

Dijkstra: How did I characterize it? “APL is a mistake, carried through to perfection. It is the language of the future for the programming techniques of the past: it creates a new generation of coding bums.” One of the things that I had told them in Munich in 1962 in an invited lecture was that programmers should not be puzzle-minded, which was one of the criteria on which IBM selected programmers, but that we would be much better served by clean, systematic minds, with a sense of elegance, yes. And APL, with its one-liners, went in the other direction. I have been exposed to more APL than I’d like because Alan Perlis had an APL period.

Frana: I didn’t know that.

Dijkstra: Oh yes. I think he outgrew it before his death, but for many years APL was “It”. And that was one of the reasons why he wrote together with Lipton and De Millo a paper against correctness proofs, because he could not do it for APL programs. He published a very obnoxious paper[10] arguing against a mathematical approach to programming.

Frana: What was wrong with ALGOL?

Dijkstra: He got dissatisfied with that or - - I’ve never understood the development of Alan Perlis, but he got severely a-mathematical. Whether that was a reaction to his environment, a set of socially acceptable… or if it was of his own, I don’t know. Okay, 1969. In that year I wrote “Notes on Structured Programming,” [11] which in retrospect I

[9] E.W. Dijkstra, A Constructive Approach to the Problem of Program Correctness,” *BIT* 8 no. 3 (1968): 174-186.
[10] Richard A. De Millo, Richard J. Lipton, and Alan J. Perlis, “Social Processes and Proofs of Theorems and Programs,” *Communications of the ACM* 22 (May 1979): 271-280.
[11] See E.W. Dijkstra, “Notes on Structured Programming,” in *Structured Programming,* ed. O.-J. Dahl, E.W. Dijkstra, and C.A.R. Hoare (London: Academic Press, 1972), 1-82.

think owed its American impact to the fact that it had been written at the other side of the Atlantic Ocean; which has two very different sides.

Frana: Yes it does.

Dijkstra: It does, yes, and I have the feeling that I can talk about this with some authority, having lived here for the better part of seventeen years.

Frana: The Netherlands and Texas are very different places.

Dijkstra: Sure. But there is one thing that one of these days I will state emphatically in public, but I can voice it already here, and that is that I think that thanks to the greatly improved possibility of communication, we overrate its importance. Even stronger, we underrate the importance of isolation.

[transition]

Dijkstra: See, look at what that ’63 invitation to the ACM workshop illustrates, at a time when I had published very little, I had not done that much either. I had implemented ALGOL 60 and I had written a real-time interrupt handler, I had just become a professional programmer. Yet I turned out to be quite well known. How come? I didn’t fully understand it, but what certainly played a role is that, thanks to my isolation, I would do other things and would do things differently than people subjected to the standard pressures of conformity.

Frana: I see, the IBM way.

Dijkstra: Yes. I had no pressure to please IBM or please the people who thought that IBM was great. I was a free man.

Frana: So IBM Europe did not have control the way …

Dijkstra: No. One of the things that saved Europe was that until 1960 or so, the first ten years so to speak; it was not considered an interesting market. So we were ignored. We were spared the pressure. Of course I didn’t understand all that at the time. I had no idea of the power of large companies. It was only recently that I learned that in constant dollars the development of the IBM 360 has been more expensive than the Manhattan Project.

Frana: I didn’t know that.

Dijkstra: Yes. Considering that the IBM 360 might have created greater damage, it would be appropriate.

Frana: Now American popular culture also, does that have an effect? The sort of homogenization of the culture that exists?

Dijkstra: I did not notice that. I remember that when I was beginning to see American publications in the first issue of the *Communications of the ACM,* I was shocked by the clumsy, immature way in which they talked about computing. There was a very heavy use of anthropomorphic terminology, the "electronic brains or machines that think."

Frana: Yes.

Dijkstra: Yes, and that is absolutely killing, because one of the most devious things we do is existing in time and the use of anthropomorphic terminology forces you linguistically to adopt an operational view. And it makes it practically impossible, at least very difficult to argue about programs independently of their possibility of being executed.

Frana: So is this why Artificial Intelligence research seemingly doesn't take hold in Europe?

Dijkstra: There was a very clear financial constraint: at the time we had to use the machines we could build with the available stuff. There is also a great cultural barrier. The European mind tends to maintain a greater distinction between man and machine. It's less inclined to describe machines in anthropomorphic terminology; it's also less inclined to describe the human mind in mechanical terminology. I don't know whether you realize that Freud never became a rage in Europe as he had become in the United States.

Frana: Maybe this is a good place to- - I was reading through some of your Burroughs reports and there was a really interesting statement there. Forgive me for repeating this back.

Dijkstra: Yes.

Frana: You said, "The tools we use have a profound and devious influence on our thinking habits, and therefore on our thinking abilities."

Dijkstra: Yes.

Frana: And then, in another place, you said, to take a different tack, "As long as we regard computers primarily as tools we might grossly underestimate their significance. Their influence as tools might turn out to be but a ripple on the surface of our culture." Are you talking here about the whole cybernetic gaze that they have lent to society? Have we become like - - do we think like our machines?

Dijkstra: No no no no.

Frana: Do our machines think like we would wish them to?

Dijkstra: No. The devious influence was inspired by the experience with a bright student. In the oral examination we solved a problem. Together we constructed the program, decided what had to be done, but very close to the end, the kid got stuck. He could not conceive of - - and actually I was amazed because he was a bright kid and he knew perfectly well, he had understood the problem perfectly. And he noticed that he got stuck and without acceptable reason. We were almost finished, he had already his grade, in my mind at least. We discussed how he got stuck. It turned out that what he had to do was to write a subscripted value in a subscript position, the idea of a subscripted subscript, something that was not allowed in FORTRAN. And having been educated in FORTRAN, he couldn't think of it, although it was a construction that he had seen me using at my lectures.

Frana: So the use of that language had made him unable to solve that?

Dijkstra: Indeed. And I have also discovered that when young students had difficulty in understanding recursion, it was always due to the fact that they had learned programming in a programming language that did not permit it. If you are now trained in such an operational way of thinking, at a given moment your pattern of understanding becomes visualizing what happens during the execution of the algorithm and the only way in which you can see the algorithm is as a FORTRAN program. So your pattern of understanding becomes familiarizing yourself with the FORTRAN program that would do it. Which, of course, meant that understanding recursion implied that he felt mentally compelled to implement recursion in FORTRAN. I did not expect him to do be able to do that.

Frana: I've heard a similar comment from a friend of mine, Herb Pelnar, a programmer on System 3 all the way through System 38; he said that it wasn't until after he retired that he really came to grips with the object-oriented approach to programming. He attempted many times to really understand it, but the way that he looked at the problems prohibited him from understanding it all.

Dijkstra: Yes.

Frana: And what's the answer then for our future students? Try to avoid the same trap?

Dijkstra: Teach them as soon as possible a decent programming language that appeals to, that exercises their power of abstraction.

Frana: Where does- - at what point then does this idea of mathematical programming become a way of describing the approach that you were taking? Was that a term you coined or was that…

Dijkstra: Mathematical programming? No.

Frana: No, it was an idea that someone else…

Dijkstra: That's a dangerous term because there are also the terms linear programming and dynamic programming, and they have nothing to do with programming. They are optimization techniques.

Frana: Never mind that mathematical programming- - it sounds like a contradiction in terms.

Dijkstra: Now that was one of the things that I learned in 1968, in Garmisch. See, I was at the Department of Mathematics at the Technical University at Eindhoven, and the official title of our product was "Mathematical Engineer." Now I had thought the design of sophisticated digital systems an area *par excellence* for a mathematical engineer, it has all the constructive aspects, it has all the proof aspects it has … Of course you must find your ways of shaping the program such that its intellectual control becomes amenable to the techniques of scientific thought. You have to do that. During Garmisch I learned that in the ears of the Americans, a mathematical engineer was a contradiction in terms: the American mathematician who is an unpractical academic, whereas the American engineer is practical but hardly academically trained.

Frana: To this day?

Dijkstra: To this day, yes. And that makes all these discussions very, very difficult. You notice that, you realize that all important words carry different, slightly different meanings. I was disappointed in America by the way in which it rejected ALGOL 60. I had not expected it. In a way I consider it a tragedy because it is a symptom of how the United States is becoming more and more amathematical. It has used all the twentieth century for doing that, as Morris Kline illustrates eloquently.[12] Precisely in the century which witnesses the emergence of computing equipment, it pays so much to have a well-trained mathematical mind. And when I mention the mathematical mind, I mean less the *quod* than the *quo modo*, less the subject matter of the considerations than the way in which the problems are approached. That's how you recognize someone being a mathematician, even if he deals with topics you have never thought about.

Frana: But the business of America is business, this country can absorb as many COCOL programmers as it can produce.

Dijkstra: Yes. NYNEX employs ninety thousand COCOL programmers.

Frana: Nynex does?

Dijkstra: At one time it did; that is at least the rumor I heard.

Frana: I wanted to ask you another question about comparing Americans and Europeans. A friend of mine, Peter Patton, is the chief technology officer at Lawson Software in St.

[12] Morris Kline, *Mathematics in Western Culture* (Harmondsmorth, Middlesex, England: Penguin Books Ltd., 1972).

Paul. He ran the Academic Computing Center at the University of Minnesota for many many years, and the first supercomputing center there. And he did installations of CDC 6600 machines in Europe. And I did not know this for quite some time, but he wrote as a fellow named Libellator in *Communications* in ’63 of his perceptions of the differences between Americans and Europeans. And he said that European programmers are fiercely independent loners, whereas Americans are team players. And that seemed to contradict, I am fairly young, my idea of how programmers behave in America and in Europe. I would have thought it would be switched around- - that Americans are fiercely independent, that the Europeans are team players.

[Pause for a Coffee Break]

Frana: Did you want to address that comment?

Dijkstra: That was very perceptive of your friend. A number of years after the war Marten Toonder, a famous Dutch artist who illustrated his own stories. He wrote one about “Heer Bommel en de teemgeest” - - “teemgeest” being an invented word - - in which he ridiculed the novel notion of “team spirit”. The comment on loners - - well, when I came to Eindhoven at the Department of Mathematics I came with ten years of experience at the Mathematical Centre, where we used to cooperate on large projects and apply a division of labour; it was something of a shock when I entered that community in which everybody worked all by himself. After we had completed the THE System, for instance, Nico Habermann wrote a thesis about the Banker’s Algorithm, and about scheduling, sharing, and deadlock prevention. The department did not like that because it was not clear how much he had done by himself and how much had been done in cooperation. And, as a matter of fact, they made so much protest that Cor Ligtmans, who should have written his Ph.D. thesis on another aspect of THE System - - he refused to do so.

Frana: For fear that he also was…

Dijkstra: Yes. It was not…

Frana: What I find interesting about this is the nature of the curriculum doesn’t reflect what you’ve said about the individualism of the student, of the programmer. There seems to be a much more general-purpose education in Europe than in America, where it seems very highly specialized. Is the outcome of the curricula different?

Dijkstra: I must be very careful with answering this because during my absence, the role of the university, the financing of the university, and the fraction of the population it is supposed to address have changed radically. That already started in the 70’s. So whatever I say about the university is out of date and also probably idealized by memory. Yes. But a major difference was that the fence around the University Campus was higher. To give you an example, when we started to design a computing science curriculum in the ’60’s, one of the firm rules was that no industrial product would be the subject of an academic course.

Frana: That sounds remarkable. I don't know what you would teach today.

Dijkstra: It's lovely. This immediately rules out all Java courses, yes, and at the time it ruled out all FORTRAN courses. We taught ALGOL 60, it was a much greater eye-opener than FORTRAN. It was the power of IBM that made the United States a backward country.

Frana: Is there a relationship between this, the curriculum and the nature of funding of universities? That is the public and private investments in education have- - we've seen that change just in the last fifteen to twenty years dramatically.

Dijkstra: Oh yes, dramatically.

Frana: Do those make computing science different? The choices you make must be different.

Dijkstra: Yes. It has the greatest influence on the funding of research projects. Quite regularly I see firm XYZ proposing to give student fellowships or something and then, somewhere in the small print, that preference will be given to students who are supervised by professors who already have professional contact with the company. Versions like that.

Frana: Are those proposals accepted at UT Austin? Or are they rejected? Modified?

Dijkstra: I don't know, but I am afraid that UT Austin is not as critical as I would have wanted. But I would like to recall the reasons why I think that it is fair to decide that computing science started with ALGOL 60.

Frana: Sure, we haven't done that. And I also want to know why computing science comes from different departments it seems in America, than in Europe. So I don't know if those conversations could be threaded together but…

Dijkstra: No.

Frana: Maybe not.

Dijkstra: Now the reason that ALGOL 60 was such a miracle was that, on the one hand, it was not a university project but a project created by an international committee, while on the other hand it introduced about a half dozen profound novelties. What it introduced, first of all, was the absence of such arbitrary constraints as, say, ruling out the subscripted subscript, the example I mentioned and that showed how harmful such things could be. A second novelty was that at least for the context-free syntax, a formal definition was given. That made a tremendous difference! It turned parsing, i.e. major aspect of compiler writing, a rigorous discipline, no longer a lot of handwaving, but perhaps more important, it made compiler writing and language definition topics worthy of academic attention. It played a major role in making computing science academically respectable. The third novelty was the introduction of the type “Boolean” as a first-class citizen. It turns the Boolean expression from a statement of fact that may be wrong or right into an expression from a statement of fact that may be wrong or right into an expression that has a value, say, “true” or “false.” How great that step was I learned from my mother’s reaction. She was a gifted mathematician, but she could not make that step. For her, “three plus five is ten” was not a complicated way of saying “false,” it was just wrong. She could only view Boolean expressions as statement of fact and not as expressions with which you can calculate without interpreting them. Potentially this is going to have a very profound influence on how mathematics is done, because mathematical proofs, which used to be something very special, can now be rendered as simple calculations that reduce a Boolean expression by value-preserving transformations to the value “true.” The fourth novelty was the introduction of recursion into imperative programming. Recursion was a major step. It was introduced in a sneaky way. The draft ALGOL 60 report was circulated in one of the last weeks of December ’59. We studied it and realized that recursive calls were all but admitted, though it wasn’t stated. And I phoned Peter Naur - - that call to Copenhagen was my first international telephone call, I’ll never forget the excitement! - - and dictated him one suggestion with the suggestion that he include it. It was something like “Any other occurrence of the procedure identifier denotes reactivation of the procedure.” Something like that. That sentence has been inserted sneakily. And of all the people who sort of had to agree with the report, none saw that sentence. That’s how recursion was explicitly included.

Frana: Was it called recursion at that time?

Dijkstra: Oh yes. The concept was quite well known. It was included in LISP, which was beginning to emerge at that time.

Frana: So it was just that you did not want to draw attention to it?

Dijkstra: We made it overlookable, yes. And F.L. Bauer would never had admitted it in the final version of the ALGOL 60 Report, had he known it. He immediately founded the ALCOR Group. It was a group that together would implement a subset of ALGOL 60, and from that subset, recursion was emphatically ruled out. A next novelty that should be mentioned was the block structure, which in a way was the first effort to make use of technical means for improved intellectual controllability of the designs. It was a tool for structuring the program, with the same use of the word “structure” as I used nine years later in the term “structured programming.” The concept of lexical scope was beautifully blended with nested lifetimes during execution, and I have never been able to figure out who was responsible for that synthesis, but I was deeply impressed when I saw it. And finally, the definition of the semantics was much less operational than it was for existing programming languages. FORTRAN was essentially defined by its implementation, whereas with ALGOL 60 emerged the idea that the programming language should be defined independent of computers, compilers, stores, etc.; the definition should define what the implementation should look like. The adjective “compile-time” should not occur in a language definition! And that separation of concerns - - now these are five or six issues that for many many years the United States has missed, and I think that is a tragedy.

Frana: And all the arguments against were misplaced …

Dijkstra: It was the obsession with speed, the power of IBM, the general feeling at the time that programming was something that should be doable by uneducated morons picked from the street, it should not require any sophistication or what have you. Yes… False dreams paralyzed a lot of American computing science.

Frana: This is something that even Fred Brooks has seen. You've read *The Mythical Man Month?*

Dijkstra: No.

Frana: Oh, you never have?

Dijkstra: No. I met him [Fred Brooks] a month ago.

Frana: So this, "Nine people can't make a baby in a month" law.

Dijkstra: Yes.

Frana: He also somewhere in the book talks about average programmers and how there are an exceptional few who are able to program. They are ten times as efficient, they are a hundred times as efficient, they are just far and away better programmers. And he does not say it as eloquently as you have, but that there is something different about certain programmers.

Dijkstra: Yes. Yes.

Frana: And I think he was forced to use average programmers because he had no option or he didn't think he had.

Dijkstra: I designed the THE Multiprogramming System, and all the synchronization, all deadlock prevention, all that stuff, was beautiful and flawless and efficient and compact and all those good things, but on industrial programming it made no impression at all. The industrial point of view rejected what we had done and the main argument was, "You made life very easy for yourself: you had only a few people, and they were first class." As I said an hour ago, I consider the design of sophisticated data systems an area of activity *par excellence* for the mathematical engineer, yes. There are few areas where it pays so well… [Would you like another cup of coffee?

Frana: No thank you.

Dijkstra: Enough?

Frana: Enough.]

Dijkstra: There are few areas where it pays so well to be sufficiently bright and well trained. This, by and large, was rejected in the Netherlands as well. I have had no influence in the teaching of programming as a tough but rewarding intellectual discipline; this view is hardly tolerated.

Frana: But you must have convinced people here when you came?

Dijkstra: Oh yes, I have convinced individuals; I have not changed the university.

Frana: Forgive me, I've forgotten now, but who made the hiring decision when you came here?

Dijkstra: That was the Department of Computer Sciences. The chairman at the time was Mani Chandy. But it was …

Frana: Well, I think that we should have lunch soon. But I did want to ask you, again, why computing science, computer science departments, in Europe - - they seem to come out of electrical engineering programs here, but not in Europe.

Dijkstra: That's correct. Now, there are a few reasons. A major one is timing. For financial reasons, Europe, damaged by World War II, was later. So American computing was earlier, the computing industry emerged earlier. The computing industry asked for graduates, which increased the pressure on the universities to supply them, even if the university did not quite know how. The net effect was that in many places, departments of computer science were founded before the shape of the intellectual discipline stood clearly out.

Frana: I think at one point you called it a cocktail.

Dijkstra: Oh yes. You also find it reflected in the names of scientific societies, such as the Association of Computing Machinery.

Frana: It's very odd.

Dijkstra: It is. It's the British Computer Society and it was the Dutch who had Het Nederlands Rekenmachine Genootschap; without knowing Dutch, you can hear the word "machine" in that name. And you got the departments of Computer Science.

Frana: Rather than computing?

Dijkstra: Rather than the department of computing science or the department of computation. Europe was later, it had coined the term Informatics. Tony Hoare was a Professor of Computation.

Frana: It’s what ’69, Informatics, Informatique?

Dijkstra: Informatique.

Frana: It’s been a riddle for some time. No one has been really able to pin it down when …

Dijkstra: I thought it was the French that pushed it. At a given moment it - - today the English prefer Information Technology, IT and Information Systems IS. I think the timing has forced the American departments to start too early. And they still suffer from it. Here, at UT, you can still observe it: it is the Department of Computer Scienc<u>es</u>.

Frana: Plural?

Dijkstra: Plural, yes. If you start to think about it, you can only laugh, but that time there were at least as many computer sciences as there were professors.

Frana: Yes. Well, should we get some lunch, a late lunch?

Dijkstra: Yes.

Frana: Thank you.