Interview Prof. Dr. Edsger W. Dijkstra, Austin,
04-03-1985
(c) Rogier F. van Vlissingen

Biographical Details

1930: Born in Rotterdam, Holland
1948-1956: Studied Mathematics and Theoretical Physics at the University of Leyden
1952-1962: Worked at the Mathematical Center In Amsterdam - first ever person on payroll as programmer
1957: Married - Justice of the peace would not accept programmer as profession for the records, so theoretical physicist used instead.
1962-1973: Professor of Mathematics at the Technical University of Eindhoven, Holland
1973-1984: Burroughs Research Fellow and Professor Extraordinarius at Eindhoven - 'Burroughs' smallest research lab was my study.'
1984: Professor and Schlumberger Centennial Chair im computer sciences at the University of Texas at Austin.

Q Given some of the drastic statements that you have made about the quality of software, is anything changing, are we progressing to better products?

A That depends, over the last fifteen years the gap between computing science and our best industries and computing practice in industry has widened. Very much. Mainly due to the fact that computing science at the universities has made such progress. Now industry is employing younger staff that has had a much better training than in the past. Over the years I have lectured on programming all over the world and the major reaction from the audience was usually: "What a pity my boss is not here.' That is a very sad comment, but the moral of the story is that if industrial practice suffers it is largely due to the conservatism, technical incompetence, of the managers of those projects. If however the internal organization and hierarchy of industrial programming organizations can be changed, there is room for great progress. And I think it will change, because the pressure from below becomes stronger and stronger.

Q In practice it often appears that pressures of

production reward clever programming over good programming: how are we progressing in making the case that good programming is also cost effective?

A Well, it has been said over and over again that the tremendous cost of programming is caused by the fact that it is done by cheap labor, which makes it very expensive, and secondly that people rush into coding. One of the things people learn in colleges nowadays is to think first; that makes the development more cost effective. I know of at least one software house in France, and there may be more because this story is already a number of years old, where it is a firm rule of the house, that for whatever software they are committed to deliver, coding is not allowed to start before seventy percent of the scheduled time has elapsed. So if after nine months a project team reports to their boss that they want to start coding, he will ask: "Are you sure there is nothing else to do?" If they say yes, they will be told that the product will ship in three months. That company is highly successful.

Q There is a perception that the spread of personal computers is bringing with it a spread of alleged computer literacy. The 'PC revolution' is thought to ready the lay user more for computer based solutions and should thus help the progress of computer use in corporations. Is there any substance to this perception or is this development maybe just muddying the water?

A Two comments to this question. One comment is that your view of industrial programs as pointed out in the question is narrow. There are all sorts of programs that hardly have users, if you think of a telephone exchange, or digital controls in cars or airplanes. As to the programming products that are used by people, I hardly have first hand experience, my impression is that an enormous amount of user time is wasted figuring out what the system does and how to control it, which is the consequence of two sorts of happenings. First of all that the designers have failed to keep the interface of a system as simple as possible - which is a challenge; but as soon as you realize that the main challenge of computer science is how not to get lost in the complexities of their own making, it is quite clear that this is a major task. Secondly, the scene is very much burdened by the fact that a large fraction of the people involved are functionally illiterate; particularly in the United States.

Q Do you really feel that this is more the case here than in Holland?

A Oh yes. In Europe a much larger fraction of the population *can write*. People in America have really suffered from the combination of TV, which makes reading superfluous, and the telephone. A few years ago I was at CalTech and that is a high quality place and everybody recommends it because the students are so bright. A graduate confessed to me - no he did not *confess*, he just stated it, that he did not care about his writing, since he was preparing himself for an industrial career. Poor industry!

Q This leads to another question: in the production of software is ....

A I prefer the term design of software, since it is an abstract procuct.

Q Ok, in the design of software is there a significant difference between what is going on in Europe and in this country?

A Yes. American computer science if practical remains much closer to the actual machine, and if theoretical is much more theoretical, whereas my personal fascination with the topic is the fact that the fruitful area of overlap of practical and theoretical is so huge. What is really nice from a theoretical point of view is usually eminently useful, and what is a really good practical idea always has something deep underlying it. And my fascination with the topic is that there is a large area where the traditional distinctions between pure and applied is meaningless. I always refuse to be called a pure computer scientist if that implies that I am unpractical. I also refuse to be called a practical computer scientist if that means that what I do is theoretically shallow, or lousy. That synthesis between pure and applied, where the distinction disappears is more readily realized in the European scene than here. Traditionally the distinction between pure and applied mathematics has always been very strong in the USA. A simple example is the fact that it is quite common in universities here to have separate departments for pure mathematics and say statistics and operations research. Such a separation would be unheard of in most European universities. Secondly, the American computer industry carries more weight, and as a result there is a tendency to see computing exclusively in an industrial context.

Q Does this difference have any bearing on the nature of products developed in Europe as opposed to in the USA - speaking of products used by people, are they more user-friendly to use that modish word?

A User-friendliness is a word that never should have been invented.

Q What word should have been invented to connotate the idea?

A The idea is wrong! It is a long story about the user, which I will try to condense. The point is that the computer user, as functioning in the development of computer products is not a real person of flesh and blood but a literary figure, the creation of literature, rather poor literature. 15 years ago I noticed that Dutch computer scientists developing products, when talking of the needs of the user would use - in the middle of a Dutch sentence - the American word user, which of course is perfectly translatable, as you and I both know. Our cigarette packages have english on them as well.....; but then I discovered that in spite of all their anglophobia, the word *user* is perfect French. Then I discovered that it is also perfect Russian and the two of us also know more Japanese than you think. Well the mere fact that that little word is not translated, but it is taken over as a foreign body, in Dutch, French, Russian and Japanese discussions, means that it has lost its original meaning. Now, if you start to analyze the many character traits of that literary figure, you discover that he is most uninspiring. He is stupid, education resistant if not education proof, and he hates any form of intellectual demand made on him, he cannot be delighted by something truly beautiful, because he lacks the education to appreciate beauty. Large sections of computer science are paralyzed by accepting this moron as their typical customer. Rare are the computer companies that are prepared to make a Mercedes, the analog the high quality product for the discerning customer. As it turns out, particularly in the USA mathematics is the pinnacle of user unfriendliness, if you read the catalogs of text book publishers, then it is quite clear that the major recommendation that they give a book is that it is amathematical, that it does not require mathematical knowledge, etc. So, user friendliness is, among other things the cause of a frantic effort to hide the fact that eo ipso computers are mathematical machines.

Q What is to be done for the industry to break that impasse?

A It may not be possible. Five or six years ago Harlan B. Mills has falsely argued that the computing industry, the computer users and the educational institutions were in a complete deadlock situation. The industry does not feel responsible for educating the people, for they are extremely conservative in their products; so that sets the standard of the demands of users for industry. Universities do not dare to teach the real stuff in their undergraduate curriculum, because the industry and the future employers are not looking for that. His conclusion was that breaking this deadlock was a federal responsibility. I think universities can. If they do not do that, they foresake their role. What is the implication of being a leading institute? What is most needed is, at a number of good universities, a few strong departments of unfashionable computer science. Whether this is planned or not, I think some professors of computer science just will not be suppressed. They can not be tamed to present a lot of junk.

Q And you intend to be one of those not suppressed.....?

A I have never been that!

Q In that fight for liberation, is your move to the US of special significance?

A No, because I carry out my work just as I did before. I got a very nice offer and the reason that I welcomed it was that, having eleven years with Burroughs Corp. - eleven wonderful years, for which I am very grateful to the company - but in course of those years my interest broadened beyond the technical needs of the computing industry. At least as the computer industry perceived them. I felt that a return to the university campus would be appropriate.

Q Is the USA the paradise for computer science that it is made out to be?

A No.

Q Why would you say that?

A Because it is in danger of being supported to death. It is certainly not a paradise for computer science. One of the things a computer scientist has to do is to distinguish between the intrinsic problems of computer science and the use of computers in society. And we should for instance clearly distinguish between problems of computer science and problems generated by an educational system. We should also clearly separate the problems of computer science and those generated by the traditional interface between the industry and its customers. User satisfaction is not a quality criterion for a computer product.

Q What would make you say that?

A I was introduced to user satisfaction as a quality criterion 20 years ago and I found it ridiculous. You can achieve it in all sorts of ways. For instance by not educating your customers; telling them that it cannot be any better. Postulating that the fact that software has bugs is a law of nature. You can even achieve it by intimidation. The worst part of it is that the goal is too fuzzy to give any technical aid. No, user satisfaction as a criterion is very typically American, because the American interface between the supplier and his customers, is that a supplier leaves the last stages of quality control to the customer. That is why you always see 'satisfaction guaranteed' and if not you may return it: they'll mend it endlessly. Something has happened with the design of programming languages. Programming languages of course have to be user friendly, programmers have to like them. For a long time features of programming languages were included on account of their supposed popularity - the main criterion was will people like it - again stemming from a rather dismal perception of the user. The great change in programming languages came from the fact that we started giving formal definitions of the semantics of programming language concepts. Something you need if you want to be able to prove things about the programs written in it. And even if you do not intend to that formal way, it was an extremely healthy exercise for programming language designers. Formalization acted as an early warning system: if the formal definition of a feature gets very messy and complicated, then you should not ignore that warning.

Q Can you explain some more about those language developments that have these better features?

A The versions that computer scientists are experimenting with are languages that are not selected for their potential appeal to the uneducated, but are screened by criteria such as mathematical power in a very rigorous sense, mathematical elegance and that kind of research produced all sorts of things. Some stay within the framework of imperative programming languages, because we know so well how to implement them very efficiently. Others are exploring functional programming languages.

Q Can you explain the difference between the two?

A I can try in a minute. Others are trying to be in their considerations even less constructive. They define the answers by a number of logical equations, leaving it to the implementation to find the solution to that set of equations. The net effect of it seems to be that a full system for really acceptable programming will be at the same time a full system that will suffice for the description of constructive mathematics. What is happening is that the gap between a program a computer may execute and the mathematical proof that the answer exists is narrowing.
The simplest way to characterize the difference between imperative programming languages and non-imperative ones is that in the reasoning about imperative programming you have to be aware of the 'state' of the machine as is recorded in its memory. In conjunction with that there is a clear distinction between a program on the one end and the information processed on the other. In the case of functional programming you create a language in which you can write all sorts of expressions and evaluating an expression then means massaging that expression until you have it in the shape that you want to have it in.

Q You are speaking of the leading edge of research of computer languages.....

A Yes, What it is again in danger of being supported to death because one of the hopes of functional programming is that in the execution of programs written in a functional style it will be easier to exploit a lot of concurrency.

Q How much of this research is at this point flowing back into industry applications?

A Little but more than you think. I know of a definitely industrial environment, to wit a huge software house

that is part of a large oil company and whose charter is primarily to satisfy the software needs of the parent company, so this is most definitely an industrial environment, where in the case of a new program to be developed they made a prototype using a functional programming language in an amazingly short time. In fact in something like better than one tenth of the time an other technique would have required. They have an implementation of this functional programming language, but not a very good one; as a result the prototype was unacceptably slow, but the experience was that it was a very important intermediate step towards the final product. So we have seen already that, though invisible in the final product, novel programming languages and implementation techniques are beginning to play a role behind the scene. This is going to have a profound impact on the software community as a whole - you see the point is that whenever you try to benefit in the development of programs from the availability of machines then obviously the first candidate for automation are those aspects of the programming process that are more or less routine jobs. As a result the really difficult stuff remains. With all our so-called programming tools the net effect is that programming becomes more and more difficult. The easy parts are automated away and the difficult parts remain and that has now reached the stage that it requires of the software developer quite a degree of mathematical sophistication. I was very amused when some time ago in a strictly industrial environment I observed a heated discussion. The discussion involved a whole bunch of so called higher order functions - higher order functions are considered too fancy to even talk about by many mathematicians, they are functions that have functions for their argument and may return a function as a value - it was in a group of industrial computer scientists, and they talked about higher order functions as if they were the most normal thing in the world.

Q So it is happening....

A It is *happening*, yes! If I were to take a number of my traditional mathamatical colleagues and expose them to this discussion, they would not have the foggiest notion of what was going on and if they did they would not believe it.

Q It sounds like the improvements in effectiveness that you speak of should get everybody's attention real fast - so again when is the impact going to be noticeable?

A You know Max Planck's answer when he was asked when quantum mechanics would be accepted by the physical community?..... "When the current generation of physicists has died!"

Q That gives us hope?

A Yes. You should realize that in *any* field the time lag, the delay between a significant scientific progress and its acceptance by the scientific community at large not to mention the moment that it finds its way into a product should be measured in generations. The reasons for dissatisfaction with the way I have been educated - on the whole I am extremely happy with my education - are that nobody warned me about that time lag of a few generations, I had to discover that all by myself with a lot of frustration.

Q What is happening to the generation of current students that you referred to before, who bring some of these concepts if not some of the products with them?

A Twenty years ago or so I came to the conclusion that the only people I could educate would become misfits.

Q Does that leave industry any hope?

A Oh yes, the point is we can educate the scientists, who educate the industry, so as to make the industry ready to employ them. The backwardness of industry is not a scientific problem. Industry is becoming aware of the problems caused by its own inertia. By definition of course this took a long time. People are reacting to it, whether it is successful or not no one knows.... General Motors wants to develop a new car in a completely new company!

Q So you are saying that this route was take to break through that inertia?

A Sure. It is a way of starting afresh.

Q On the matter of cooperation between industry and university research, do you feel that things are done better here or in Holland. Or can both learn from each other?

A That is hard for me to judge, because since I am here I have had no experience with the cooperation with industry. One of the reasons in '73 for my joining Burroughs Corp. as a research fellow was that in the preceding decade I had observed that it was extremely hard to penetrate industry from a campus, from outside. In the next ten years I discovered it was hard from the inside as well, though easier. This applies to industry on both sides of the Atlantic. It is definitely so that industry here is more used to giving support to individual research - that has always been the case. It is hardly possible to be a professor and not also be a consultant. But many consultants do their own thing.

Q Do you feel that Holland should learn from this American approach?

A My overall impression is that it does not matter what mistakes the USA makes, because they will be faithfully copied in Europe only 20 years later. Europe can learn an immense amount from the United States. By studying what is happening and then not copying it. In some aspects we might even try to copy it.

Q What are you thinking of specifically?

A Oh, the ease with which you can incorporate, I believe it costs $3 or so.....

Q Considering possible breakthroughs, are we at this moment constrained by technology, by programming techniques, or maybe both?

A John McCarthy of Stanford University has made the remark a long time ago and it is still very valid, that the greatest bottleneck is not to be found in hardware but in our programming ability.

Q Speaking of programming bottlenecks - what will the impact of the research in artificial intelligence be?

A Can you research something that is not science? I feel that the effort to use machines to try to mimic human reasoning is both foolish and dangerous. It is foolish because if you look at human reasoning as is, it is pretty lousy; even the most trained mathematicians are amateur thinkers. Instead of trying to imitate what we

are good at, I think it is much more fascinating to investigate what we are poor at. It is foolish to use machines to imitate human beings, while machines are very good at being machines, and *that* is precisely something that human beings are very poor at. Any successful AI project by its very nature would castrate the machine.

Q Computer science has become very popular, are there too many students, are there too few students - and do you see any change in the level of preparedness of the entering student between now and say ten years ago?

A The topic is devastatingly popular. Because computing is now supposed to cure all the ills of the world and more. I once gave a short summary of the fact that over the centuries scientific effort had been a complete disaster. The first scientic effort of course was the production of the elixir that would give eternal youth, but very rapidly they discovered that there was little point in living eternally, if you would live in eternal poverty, so the next major scientific project was how to turn anything into gold. Now it was quite clear that the planning of these two major research efforts was beyond the powers of the seers of the day, so for sound managerial reasons the next hot issue became the accurate prediction of the future. As time went by the original goals were forgotten. Medicine divorced itself from quackery, chemistry divorced itself from alchemy and astronomy divorced itself from astrology. However there is still some feeling of guilt in the academic community, because as soon as promising new branch of science or technology emerges, it is saddled up with the old hopes. The current boom in computing is an immediate reflection of absolutely unrealistic hopes. So if you ask whether there are too many or too few students: *an order of magnitude too many.* From a scientific point of view you would like to weed out the lot. Keep the brightest 2% and do business. The current generation of freshmen coming in is not only ill prepared, they have been misguided.

Q Are you suggesting that the personal computer boom has fostered fruitless play rather than ....

A Yes. I have said it before in public and I am perfectly willing to repeat it that someone introduced to computers via Basic is in all probability mentally mutilated beyond redemption. That is no joke. A major branch of the Siberian Academy of Arts and sciences is

aimed at keeping Basic out of Siberian high schools.

Q So what will get in?

A Probably Basic.

Q Of other languages that are becoming popular, are there any that offer any better hope?

A Oh yes. The popularity of Pascal is very encouraging. Not because it is ideal. but it is definitely orders of magnitude better than any of its competitors. Besides that it is interesting because it is a one man product, without any form of political or industrial backing. It has gained its popularity in slightly over a decade, because at the time of its inception it was so much better than anything else available.

Q We have seen Lisp emerge in documenting programming constructs?

A Lisp was great at the time of its inception if only for a radically new way of using machines. It has suffered the fate of having been elevated to a status of de facto standard with all its defects. Despite its conceptual novelty a lot of the design of Lisp is terribly amateurish even by the standards of the time. When I read the first Lisp manual, the Lisp 1.5 manual, published in 1961, I could not believe my eyes. It was an extremely poor language. Now it has become the defacto standard of the AI community, which now suffers from Lisp, the way the rest of the world suffered from Fortran.

Q Is the field of computer science approaching maturity in any sense, or where are we?

A Well, we is a rather mixed lot aren't we? Let me give you three dates 1968, 1975 and 1984.
In 1968 IBM published an ad in Datamation showing Suzie Meyer smiling in full color. Suzie Meyer announced that she had solved all her programming problems by the simple trick of switching over to PL/1. It is a great pity that Datamation did not publish a picture of Suzie Meyer four years later!
1975 was about the time that the Swedish logician Per Martin-Loeff convinced himself of the fact that a well documented program was an object logically isomorphic

with a constructive mathematical proof. In that sense the word constructive refers to the most puritan of all mathematics.
By 1984 the computer science community had taken notice of that fact. Late 1983 I attended a small industrial conference on programming. One message came across loud and clear and that was that in development of programs formal techniques would not only be indispensible, but would have to be applied on a scale without precedent. Now if you compare that with Suzie Meyer then we have travelled long and far.

Q As a final question I would like to ask you what is the most interesting research project that you see right now if you had to name one?

A I can name only one but it is in very broad terms. It is how to increase our powers of reasoning by orders of magnitude and how to apply as part of that formal techniques on a scale without precedent. My firm belief is that the potential impact of the computer on mathematics in general will be as profound in the century to come as the influence of physics on analysis has been in the preceding century.