Edsger Dijkstra

Brian Randell

School of Computing Science, University of Newcastle upon Tyne, UK

Brian.Randell@newcastle.ac.uk

1. Introduction

I was most honoured, but also somewhat confused, by the invitation to give a banquet speech about the late great Edsger Dijkstra at WORDS 2003. This is because the term "banquet speech" is American, and translating it into the English term "after-dinner speech" is rather misleading. In Britain an "after-dinner speech" is meant to be, above all, humorous, so as to be capable of entertaining, and retaining at least the attention, if not the enthusiasm, of an audience that have just eaten and drunk, to excess in all probability.

But the privilege of talking to you about Edsger Dijkstra is not an occasion for humour – or at least only occasional humour, in the very personal portrait I'm going to try and give you of one of computer science's intellectual giants, one I am honoured to have called a friend.

I have another cause for misgivings. What I can say about Dijkstra is bound to seem inadequate to any of you who knew him – and will feel to me to be inadequate for any who didn't. But with these caveats, let me begin.

You have, I believe, all been given a copy of a leaflet [Campbell-Kelly 2002] about Dijkstra – I hope you've had a chance to read it. My aim is not to relate all the factual statements it contains, which between them amply testify to the extent and importance of his contributions to computer science, but rather to augment these statements with some mainly personal recollections – and also to encourage all of you to explore for yourself some of his work through his many writings.

I should explain the background to this leaflet. It was produced for a British Computer Society meeting last year honouring the memory of Edsger Dijkstra, a meeting at which Tony Hoare and I were the main speakers. I used the invitation to speak at the meeting as an opportunity to tell Ria Dijkstra (Edsger's widow) and a lot of people who knew Dijkstra well, or at least knew lots about Dijkstra, what a debt I owed to him. I will use here some of the remarks I made on that occasion, but I will include rather more information about Dijkstra's work and its significance than I did in my brief talk at the BCS meeting.

2. Brighton 1960

For me, the story of Edsger Dijkstra begins in 1960, when I heard him lecture on his Algol 60 Compiler at a conference in Brighton.

I hope I don't have to convince you of the importance of the Algol 60 Report [Naur (ed.) 1960], both for the Algol language, and as an example of how to define a programming language, in particular its syntax. The Report documented the work of a committee, but was itself largely the work of Peter Naur. Dijkstra was at this time a programmer at the Mathematical Centre, Amsterdam, whose Director, Professor van Wijngaarden, was a member of the Algol Committee.

Within seven months of the publication of the Report, Dijkstra and a colleague, Jaap Zonneveld, had completed a compiler for virtually all of Algol 60, despite the fact that language had many highly novel features, and great generality. (For example, it had recursive procedures, hierarchical name scoping, multi-dimensional arrays whose size could depend on input calculations, powerful parameter passing mechanisms, etc. – facilities far exceeding in generality and elegance those of the two main languages that predated it, Fortran and Cobol.) Indeed, Dijkstra and his colleague produced their compiler at a time when no-one on the committee had yet figured out how to compile the language whose facilities they had just approved.

At this time I was at the English Electric Company, at Whetstone, a small place outside the city of Leicester (or "Lie-cester" as I should pronounce it – since this is after all an American-style banquet talk). I was managing a small "Automatic Programming Section", which had been established after I and a colleague had – as a bootleg project – produced a compiler for English Electric's first computer, the DEUCE. (The first DEUCE was produced in 1955, but its design was very closely based on Alan Turing's original plans for the ACE computer at the UK's National Physical Laboratory [Turing 1945]. Hence its order code was extremely cleverly designed, but usable well only by extremely clever programmers, i.e. by the likes of Alan

Turing himself – hence the need for a compiler, for use by ordinary mortals.)

Another colleague, Lawford Russell, and I were then planning – for an as yet unspecified language – a compiler aimed at easing the task of developing programs for English Electric's new computer, the KDF9 [Davis 1960]. After Dijkstra's talk, someone suggested that Lawford and I should target our work on Algol, and seek Dijkstra's assistance. This led – eventually, since it took quite a while to get permission – to my first ever expenses-paid foreign travel!

3. Visit to Mathematical Centre

Lawford and I spent a wonderful, but highly intensive week in Amsterdam – learning about the Electrologica X8 compiler – and discussing plans for a compiler for KDF9, a compiler that later became known as the Whetstone Compiler. This was to be a compiler for use during program development, there being plans elsewhere in English Electric for an optimising compiler for the machine. Hence we decided that we would produce a compiler that would convert Algol into an intermediate (Pcode-like) language. In effect we designed a computer architecture that would be convenient to compile for and that would be implemented by an interpreter.

Many details, small and large, of that wonderful week are burned into my memory. For example, I recall that the Mathematical Centre carefully booked us into the Hotel Krasnapolsky – they knew from experience that it was virtually impossible for a foreign visitor to Amsterdam to mispronounce the name so badly as to confuse a taxi driver.

We spent each morning and afternoon in intensive discussions with Dijkstra, but he asked us to look after ourselves at lunchtime, so that he could have a rest from speaking English, though in fact his command of the language was already impressive. Indeed, during a very pleasant evening with Edsger and his wife Ria in their apartment, I recall him seeking linguistic help only with some of the more abstruse wording of a song by Tom Lehrer.

4. Our Trip Report and Book

After returning from Amsterdam we produced a detailed report documenting these discussions [Randell and Russell 1962]. This was thus both a description of Dijkstra's original compiler, and an initial account of how our planned compiler for the KDF9 might work. To our pride (and now to my regret) Edsger used the availability of our report both to fend off further would-be visitors, and as an excuse to avoid producing his own detailed account of his compiler.

I've been referring to him as Edsger, but in fact during all this time, and in the correspondence that followed our visit, we were still addressing each other formally. (This was Europe, and the 1960's!)

But then Lawford and I found a way of usefully improving Dijkstra's compiler, which he had designed simply to report the first error it found in any Algol source program and then stop. (This was in line with his attitudes to programming and programmers, but not very appropriate for the kind of industrial program development environment that we had in mind.) He in fact had a horror of producing a compiler that simply proceeded regardless with the compilation after detecting a source program error, and so – almost inevitably – got confused and 'discovered' all sorts of spurious errors. However, we came up with a scheme that almost entirely avoided producing any incorrect error reports while going on and looking for further actual errors in the source program. We documented this in a report, and sent it to Dijkstra. His reply, for the first time ever, started 'Dear Brian' – I felt as if I'd just been awarded a higher degree.

We continued to correspond with Dijkstra as we designed our compiler. In retrospect, it is now clear that we greatly benefitted from an annoying series of delays to the completion of the first KDF9 computer – a period during which we refined our design and other people started using it to develop compilers for their own computers. Then someone startled us with the suggestion that we write a book describing our compiler. We of course asked Dijkstra what he thought of this. He was very supportive, and gave us some very valuable, albeit uncomfortable, advice. This was that instead of just describing our compiler, we should try to list all the alternatives we had considered for each design choice and explain the basis on which we had chosen amongst these alternatives. Moreover, he advised that we should make a point of admitting it when our choice had been arbitrary and/or had in retrospect proved to be wrong. I have always been extremely grateful for this advice – and have taken care to pass it on to all my graduate students.

Edsger's advice to me was of course in line with the fact that he always himself set and followed very high standards for clarity and presentation of writing and lecturing. Although the lecturing style that he developed had its critics (some of whom mistakenly interpreted his reflective pauses as mere theatricality), my own regret is that I cannot match either the clarity with which he lectured, or the skill he demonstrated throughout his career at inventing or choosing appropriate problems and examples.

5. 1964 – IBM and Algol WG 2.1

By 1964, our Whetstone KDF9 compiler, and a number of other versions of it, had been completed, and our book had been published [Randell and Russell 1964]. To my surprise, I was invited to join the IBM Research Center, in Yorktown Heights, New York. (I had not thought of what we were doing as research – we produced

the compiler because it was needed.) With some misgivings, which I'm sure Dijkstra shared, since to him IBM was 'The Great Satan', such was his dislike of their hardware and software, I joined IBM – and thus was able at last to take up my invitation to membership of the IFIP Working Group 2.1 on Algol. (I hadn't been able to get permission from English Electric to accept – one trip abroad was enough in their view!). By this time Dijkstra was also a member – and from then on, for many years at the Algol committee and elsewhere, I had numerous opportunities to meet him – and to have numerous memories to draw on for this talk.

When I joined IBM I made it clear I had no intention of writing another compiler. Instead I got involved in compiler architecture and in particular in operating systems – as had Dijkstra. Indeed before he got involved in compilers, he had worked on communications facilities, and (the entirely novel, and to his view of programming, fearsome idea of) interrupt handling. This was for the Electrologica X1 compiler, work for which he obtained a PhD – work that provided a basis on which he laid many of the foundations of the entire subject of concurrent programming.

6. 1967 SOSP

Dijkstra's seminal contributions in the area of concurrent programming were on problems such as process synchronisation, critical sections, and system deadlocks – for example, his famous work on the "dining philosophers" problem. He, and this work, starred at the ACM's first Symposium on Operating System Principles, where he presented a paper on the THE Operating System - see below. I also attended this conference, which was held in Gatlinburg, Tennessee, at the time a so-called "dry" town, in which the only alcohol that could be obtained was very appropriately called "near beer" – whose alcohol content was as low as its temperature, and as Dijkstra's spirits when he found that it was all that was available.

At the Symposium my recollection is that Edsger used a very simple and elegant diagram (representing what he termed "progress space") showing the dangers of deadlock between a pair of processes competing for unsharable resources – the term he used at this time for deadlock was "deadly embrace", and his impact was such that the symposium organisers made an impromptu award to him, of two toy "Smoky Mountain Brown Bears" locked in a deadly embrace. But he also had one other major impact on at least some of the conference attendees – and this was quite unintended.

There had been a move to set up a committee to define an ideal list of operating system primitives. Edsger and I had both refused invitations to join this committee, with some alacrity. That evening he and I had dinner together, during which we had great fun drawing up a spoof list of desirable operating system facilities – a list that I know Dijkstra carried around in his wallet for some years afterwards. (The only item I can remember from the list is "Execute Operator.") Edsger was so pleased with our list that he took it across to a group of conference attendees at a table on the far side of the dining room, only to return looking somewhat abashed – it turned out that this group was the committee, having its first meeting!

After Gatlinburg, Edsger travelled back with me to IBM Research, where I had arranged that he would give a lecture in the large and impressive research auditorium. After his talk I confessed to him that I had planned a practical joke, but that at the last moment I had had cold feet, and not gone through with it. This was to place across the top of the lectern, invisible to the audience and to Dijkstra until he came up to the lectern following my introduction of him, a plastic strip I had taken from the hotel room in Gatlinburg. The strip said "This has been sanitized for your protection." Such a warning would have been very appropriate, given his attitude to IBM, but when I afterwards confessed what I had been planning, he thanked me for my restraint, saying that it would have taken him the whole time allotted to his lecture to stop laughing.

7. The THE operating system

Dijkstra's paper on the THE Operating System from the Gatlinburg symposium was published in the CACM in 1968 [Dijkstra 1968b]. (THE stands for "Technische Hogeschool (Technological University), Eindhoven", where he had taken up a professorship some years earlier.) The THE system was a staggering achievement, because it was so well designed and coded that there were virtually no bugs in it. One of the keys was its structure as a set of levels of abstraction, each of which hid some aspects of the underlying hardware and introduced further convenient facilities for use by user programs. But the choice and ordering of these levels were not arbitrary – rather what was critical was that at each successive level, the time granularity became roughly an order of magnitude larger. Thus one could to a great extent design these levels independently of each other.

All this was totally revolutionary! For years it was almost obligatory, in any new paper on operating system design, to reference this paper – now I fear there are generations who have never heard of it. If you are a member of one of these generations let me urge you to go find it in the CACM, and read it.

8. GoTo Considered Harmful

One of the earliest influential attempts at turning programming into a respectable discipline, rather than a mere craft, was the movement, led mainly by Dijkstra and a few colleagues, promoting 'structured programming'. At the time some people equated structured programming merely with mechanically avoiding the use of **go to** statements. Indeed Dijkstra's 1968 letter to the editor of the Communications of the ACM [Dijkstra 1968a] on this topic caused a huge debate, though it gets forgotten that

the title '**go to** considered harmful' was dreamt up by the editor rather than by Edsger himself.

It is in fact illuminating to recall the elegant argument that Edsger advanced against the use of **go to**s. The essence of this argument was that one could only interpret the value of a program variable if one could identify the present stage of progress through the computation. One could make such identifications very easily when using well-disciplined control structures such as **if** statements, **for** statements and procedure calls. However, Dijkstra pointed out that once **go to** statements were being used, one had to use the values of variables to identify the stage of progress, i.e. to understand the values of variables – clearly a most undesirable state of affairs

9. The 1968 NATO S/W Engineering Conferences

In October 1968 Edsger and I participated in the first of the NATO Software Engineering conferences, held at Garmisch-Partenkircken, in Germany [Naur and Randell 1969]. We have both since gone on record as to how the discussions at this conference on the "software crisis", and on the potential for software-induced catastrophes, strongly influenced our thinking and our subsequent research activities. In Edsger's case it led him into an immensely fruitful long-term study of the problems of producing high quality programs. In my own case, it led me to consider the then very novel, and still somewhat controversial, idea of design fault tolerance. Suffice it to say that our respective choices of research problem suitably reflect our respective skills at program design and verification.

10. Algol 68

1968 was also a critical year in the life of the IFIP Algol Committee, which was striving to develop a successor to Algol 60. By this time, Niklaus Wirth had left the committee to pursue his own ideas on language development, work which later resulted in the Pascal language. Dahl and Nygaard had produced Simula, (an extension of Algol 60 aimed at simulation) and then Simula 67, their 'Common Base Language' [Dahl and Nygaard 1967], which of course was the foundation of what became known as object-oriented programming. The Algol Committee had become dominated by Aad van Wijngaarden, and was pursuing two issues in parallel – language development, and techniques of language description – with van Wijngaarden pushing his ideas on 'two-level grammars'. A series of ever more fractious meetings was held, culminating in one in Munich in late1968, when a majority of the committee voted to approve the Algol 68 Report [van Wijngaarden, Mailloux et al. 1969]. However Dijkstra was one of the leaders of a gang of eight (which I am proud to have been a member of) that produced a Minority Report, arguing that the Algol 68 Report should not be approved [Dijkstra et al 1970]. In fact I recall we felt that Simula 67 was a more fruitful development – to my subsequent regret we did not actually say this in our Minority Report. The committee then split, and a group of us left to found a new IFIP Working Group on Programming Methodology, though we regarded this group, WG 2.3, as having twin roots, the Algol Committee and the NATO Software Engineering Conference.

WG 2.3 became an important forum, one at which Dijkstra tried out, polished and promoted many of his highly influential ideas on methods of developing high quality programs (I remained with this working group for a number of years, though later left when my interests turned elsewhere.)

11. 1969 NATO Conference

There was a further, much less successful, NATO Conference on Software Engineering in 1969 in Rome, which Edsger and I both attended [Buxton and Randell 1970]. I've mentioned that Edsger could give very clear and elegant lectures. However, he was rarely willing to make any concessions to his audience, for example in adopting any of the terminology that they might already be familiar with, and so his lectures were not always well-appreciated. This was the case in Rome, and one attendee in particular, Tom Simpson of IBM Houston, justifiably famous within IBM for his pioneering work on operating systems, in particular a system called HASP, started an argument with Dijkstra. This argument continued at intervals throughout the conference – and I watched with fascination as Tom and Edsger gradually learnt how to communicate with, and to gain great respect for, each other. It is a pity that some of the other people who then, and in the years to come, reacted negatively to Dijkstra, did not have the time or inclination to get to appreciate him properly. I'll return to this point later.

12. In 12 months from June 1974

I've talked a lot about one very eventful year in my (and Dijkstra's) life, namely 1968. However at last year's BCS meeting in memory of Dijkstra, Tony Hoare chose another twelve-month period in Dijkstra's life to illustrate how amazingly creative he was. This was the period from June 1974, which Tony illustrated by referring to a sample of the memos produced in this period.

Dijkstra produced well over a thousand EWD memos as he called them (now nearly all now on the web), many handwritten, and distributed Russian "samizdat" style for years, by a world-wide network of colleagues. (So prolific was he at least one person, not noticing that they were Edsger's initials, wrote to him asking if 'EWD' stood for 'Eindhoven Working Document'.) These are now all archived, and made available at http://www.cs.utexas.edu/users/EWD/.

The EWDs that Tony highlighted were as follows:

EWD418 – on "Guarded commands, non-determinacy and a calculus for the derivation of programs" – this led

to his book "A Discipline of Programming" [Dijkstra 1974a]

EWD426 – "Self-stabilizing systems in spite of distributed control" – which created a whole subculture of computing science, addressing the issue of how certain kinds of system might be designed so as to be inherently capable of recovering after errors have occurred and propogated far and wide [Dijkstra 1974c].

EWD464 – "A new Elephant Built from Mosquitos Humming in Harmony" – this launched the whole new class of highly parallel algorithms now known as "systolic algorithms", since data pulses through them rhythmically, rather like blood flows through the heart [Dijkstra 1974b].

EWD492 – "On-the-fly garbage collection" – again a paper which prompted a whole series of follow-ups by others, addressing the incredibly tricky problem of identifying and gradually catching up with the production of garbage (data items that had become unlinked and so no longer accessible) without stopping the system in order to tidy it up [Dijkstra 1975].

This truly is an astonishing output, in just *one* year - yet these are just four of the over *seventy* EWDs that Edsger wrote and circulated during this period.

The 60th Birthday Salute

In 1989, as Edsgers's 60th birthday approached, I and a number of his friends and colleagues, were approached and asked to provide contributions to a book that would be produced as a surprise in his honour [Feijen, van Gasteren et al. 1990]. I was concerned that my field of research, on fault tolerance and, in particular, my attitude that design faults had to be tolerated, were so far from his interests and attitudes that a contribution from me would not be appropriate. I was very kindly assured by the Editors that this was not at all the case, but was invited and agreed instead to write a Foreword to the book. This was a naive choice on my part, since this was much harder to write. However, I'd like to end this talk, first by repeating my acknowledgment of the great debt I owe to Dijkstra, and then by using some of the remarks I made in this Foreword:

> Edsger is, with all that the word implies, a perfectionist, who expects as much of his listeners and readers as he demands of himself. His programming and his mathematics are strongly guided by his concern for clarity of notation and exposition, and indeed for what he quite justifiably terms 'beauty'. Thus his descriptions of problems and solutions, both in his lectures and published papers, and in his EWD series of documents . . . are often vivid and compelling.
>
> Although over the years much of his work has had a very immediate impact, on debate if not always in practice, some of his more recent diagnoses and prescriptions have proved harder to take. This is in part because he is sometimes more concerned with the truth of his arguments than with whether they are couched in terms that will help to ensure that they have the desired effect on his audience. Nevertheless, careful study of all his writings is highly recommended to all who care for the future health of computing science.
>
> Much of the last phase of his career, at the University of Texas, where he went in 1984, was spent investigating the effective structure of logical arguments, applied to mathematics as much as programming – a distinction whose validity he denied, since to him programming *was* mathematics. This work set standards that many cannot even recognise, let alone aspire to. However I am confident that it will eventually have deep and long-lasting effects, much of it indirect, through the inspiration that it provided to close colleagues.

Let me end with a quotation from George Bernard Shaw [Shaw 1903]:

> The reasonable man adapts himself to the world: the unreasonable one persists in trying to adapt the world to himself. Therefore all progress depends on the unreasonable man.

Edsger W. Dijkstra is, in many ways, just such a man. Needless to say, the world of computing science could well do with many more "unreasonable" men (and women) of his calibre!

References

[Buxton and Randell 1970] J.N. Buxton and B. Randell, (Ed.). *Software Engineering Techniques: Report on a Conference sponsored by the NATO Science Committee, Rome, Italy, 27th to 31st October 1969,* Brussels, Scientific Affairs Division, NATO, 1970, 164p.
http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1969.PDF
[Reprinted in Software Engineering: Concepts and Techniques (eds. J.M. Buxton, P. Naur and B. Randell) Petrocelli/Charter, New York, 1976.]

[Campbell-Kelly 2002] M. Campbell-Kelly. *A Tribute to Edsger W Dijkstra,* The British Computer Society, Advanced Programming Specialist Group, Computer Conservation Society, 2002, 8p.

[Dahl and Nygaard 1967] O.-J. Dahl and K. Nygaard. *SIMULA 67: Common Base Definition,* Norwegian Computing Center, Oslo, Norway, 1967.

[Davis 1960] G.M. Davis, “The English Electric KDF9 Computer System,” *Comp. Bull.*, vol. 4, no. 3, pp.119–120, 1960.

[Dijkstra et al 1970] E.W. Dijkstra et al, “News Item—Minority Report,” *ALGOL Bulletin*, vol. AB31.1.1 (March), 1970.

[Dijkstra 1968a] E.W. Dijkstra, “Go To Statement Considered Harmful,” *Comm. ACM*, vol. 11, no. 3, pp.147-148, 1968a.

[Dijkstra 1968b] E.W. Dijkstra, “The Structure of the THE Multiprogramming System,” *Comm. ACM*, vol. 11, no. 5, pp.341-346, 1968b.

[Dijkstra 1974a] E.W. Dijkstra. *Guarded Commands, Non-Determinacy and a Calculus for the Derivation of Programs,* EWD418, Jun. 1974, 1974a. http://www.cs.utexas.edu/users/EWD/ewd04xx/EWD418.PDF
[Published as: Guarded Commands, Non-Determinacy and Formal Derivation of Programs, *Comm. ACM*, 1975, 18, 8, pp. 453–457]

[Dijkstra 1974b] E.W. Dijkstra. *A New Elephant Built From Mosquitos Humming in Harmony,* EWD464, Nov. 1974, 1974b.
[Published in: *Selected Writings on Computing: A Personal Perspective*, Springer-Verlag, 1982, pp. 79–83]

[Dijkstra 1974c] E.W. Dijkstra. *Self-Stabilizing Systems in Spite of Distributed Control,* EWD426, Jun. 1974, 1974c. http://www.cs.utexas.edu/users/EWD/ewd04xx/EWD426.PDF
[Published; *Comm. ACM,* 1974, 17, 11, pp. 643–644]

[Dijkstra 1975] E.W. Dijkstra. *On-The-Fly Garbage Collection: An Exercise in Multiprocessing,* EWD492, Apr. 1975. http://www.cs.utexas.edu/users/EWD/ewd04xx/EWD492.PDF
[Published as: Dijkstra, E. W., Lamport, L., Martin, A. J., Scholten, C. S., and Steffens, E. F. M., On-The-Fly Garbage Collection: An Exercise in Cooperation, *Comm. ACM*, 1978, 21, 11, pp. 966–975]

[Feijen, van Gasteren et al. 1990] W.H.J. Feijen, A.J.M. van Gasteren et al, (Ed.). *Beauty is Our Business: A Birthday Salute to Edsger W. Dijkstra,* Springer-Verlag Texts and Monographs in Computer Science, 1990.

[Naur (ed.) 1960] P. Naur (ed.), “Report on the Algorithmic Language ALGOL 60,” *Comm. ACM*, vol. 3, no. 5, pp.299-314, 1960.

[Naur and Randell 1969] P. Naur and B. Randell, (Ed.). *Software Engineering: Report on a Conference sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th October 1968,* Brussels, Scientific Affairs Division, NATO, 1969, 231p. http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF
[Reprinted in *Software Engineering: Concepts and Techniques* (eds. J.M. Buxton, P. Naur and B. Randell) Petrocelli/Charter, New York, 1976.]

[Randell and Russell 1962] B. Randell and L.J. Russell. *Discussions on ALGOL Translation at Mathematisch Centrum,* W/AT 841, Atomic Power Division, English Electric Co., Whetstone, Leics., 1962.

[Randell and Russell 1964] B. Randell and L.J. Russell. *Algol 60 Implementation,* Academic Press, 1964. ISBN: 12-578150-4

[Shaw 1903] G.B. Shaw. *Man and Superman - "Maxims for Revolutionists: Reason",* 1903.

[Turing 1945] A.M. Turing. *Proposals for the Development in the Mathematics Division of an Automatic Computing Engine (ACE),* Report E882, National Physical Laboratory, 1945. [Reprinted with foreword by D.W. Davies, NPL Report Comm. Sci. 57, April 1972.)]

[van Wijngaarden, Mailloux et al. 1969] A. van Wijngaarden, B.J. Mailloux et al, “Report on the Algorithmic Language ALGOL 68,” *Numerische Mathematik*, vol. 14, pp.79–218, 1969.