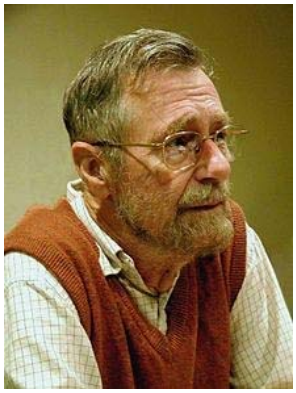
Edsger W. Dijkstra

Born 1930, Rotterdam, The Netherlands; Leading critic of programming without a mathematical proof of correctness and condemner of the infamous GOTO; recipient of the 1972 ACM Turing Award.



Education: MS, mathematics and theoretical physics, University of Leiden, 1956; PhD, computing science, Municipal University of Amsterdam, 1959.

Professional Experience: professional programmer, Mathematisch Centrum, 1952-1962; professor of mathematics, Eindhoven University of Technology, 1962-1984; research fellow, Burroughs Corp., 1973-1984; Schlumberger Centennial chair, computer sciences, University of Texas at Austin, 1984-present.

Honors and Awards: fellow, Netherlands Royal Academy of Sciences; distinguished fellow, British Computer Society, 1972; ACM Turing Award, 1972; IEEE Computer Society Pioneer Award, 1980.

Dijkstra, recipient of the 1972 ACM Turing Award, is known for early graph-theoretical algorithms, the first implementation of Algol 60, and the first operating system composed of explicitly synchronized sequential processes. He is also credited with the invention of guarded commands and of predicate transformers as a means for defining semantics, and programming methodology in the broadest sense of the term. In recent years he has been deeply involved in the applications of mathematical proof techniques to programming and the development of programs from mathematical axioms.

At the 1972 ACM Turing Award ceremony M. Doug McIlroy read the following citation:

> The working vocabulary of programmers everywhere is studded with words originated or forcefully promulgated by E.W. Dijkstra: “display,” “deadly embrace,” “semaphore,” “go-to-less programming,” “structured programming.” But his imprint on programming is more pervasive than any catalog of jargon can indicate. The precious gift that this Turing Award acknowledges is nothing less than Dijkstra's style-his approach to programming as a high intellectual challenge; his eloquent insistence and practical demonstration that programs should be composed correctly, not just debugged into correctness; and his illuminating perception of problems at the foundations of program design. He has published about a dozen papers, both technical and reflective, among which are especially to be noted his philosophical addresses at IFIP (1962, 1965a), his already classic papers on cooperating sequential processes (1965b, 1968a), and his memorable indictment of the go to statement (1968b). An influential series of underground letters by Dijkstra have recently been published in monograph on the art of composing programs (1971).
>
> We have come to value good programs in much the same way as good literature. And right at the center of this literary movement, creating, and reflecting patterns no less beautiful than useful, stands E.W. Dijkstra.

His interests focus on the formal derivation of programs and the streamlining of the mathematical argument. His publications represent only a minor fraction of his writings—he writes, in fact, so much that he cannot afford the use of time-saving devices such as word processors. He owns, however, several fountain pens, three of which are Mont Blancs, for which he mixes his own ink. His writings, which include technical papers, trip

reports, and essays on various topics, are distributed in an informal distribution tree to many colleagues. The latest is “numbered” EVVD1070, which is an indication of how prolific he has been.

In 1964 at a WG 2.3 meeting in Baden bei Wein, van Wijngaarden was showing how to get rid of GOTOs by replacing them with recursive procedure calls that never return. Dijkstra's reaction to this academic result was to spend the evening deriving programs that had no GOTOs in the first place. On the terrace during next morning's coffee break, and throughout the day, he peddled his new style, pointing out that many programs became simpler in the process and that none got harder. He invented a LOOP-EXIT statement to solve some structural problems. In less than 24 hours, Dijkstra had converted a sterile academic exercise into a movement that would shake the field when the eruption came later in 1968 with his famous letter on “the GOTO considered harmful.[1] Most people regard his 1968 letter as the start of the eliminate-the-GOTO movement; few realize that it began as a reaction to an academic talk in 1964.

Perhaps the measure of this man is best expressed by those he has influenced, and who more influenced and better placed than his students? This influence has been caused by his particularly perceptive and brilliant mind, his intense desire to be professionally honest, a discipline that is unequaled, and a way with the pen (in both form and content) that others would kill to attain. His ability to make a decision on technical grounds and then to put it into practice is unrivaled. He seems to have been endowed with all the good qualities one would like to see in a scientist, and he has taken care to sharpen them. On the occasion of his 60th birthday, the University of Texas organized a celebratory symposium on the “Frontiers of Computing” primarily staffed by his disciples. This was an occasion to provide insights of the person behind the facade of the “professor.” Even Dr. W.S. Livingston, vice president and dean of graduate studies of the University of Texas at Austin, could not resist relating his “Dijkstra experience”:

> In 1983 Dr. Dijkstra was being interviewed by the University of Texas at Austin to determine his suitability for appointment to the distinguished position. Someone decided that it was my task as vice president (even though I am by trade a political scientist) to conduct an interview on behalf of the University administration. I was not quite certain just what topics we might discuss, but Professor Dijkstra soon solved that problem. After some very short preliminaries he stood up and provided me with a lecture on his thoughts on the subject, striding up and down in my office. It is not at all clear to me just who interviewed whom.

Tony Hoare, himself a Turing Award winner and pioneer (but not a student of Dijkstra's), told of their first meeting, which exemplifies the discipline of programming that Dijkstra espoused (see Dijkstra 1976):

> The first time I visited Edsger in Eindhoven was in the early Seventies. My purpose was to find out more about the THE operating system,[2] which Edsger had designed. In the computing center at which the system was running I asked whether there was really no possibility of deadlock. “*Let's* see” was the answer. They then input a program with an infinite recursion. After a while, a request appeared at the operator's console for more storage to be allocated to the program, and this was granted. At the same time they put a circular paper tape loop into one of the tape readers, and this was immediately read into buffer file by the spooling demon. After a while the reader stopped; but the operator typed a message forcing the spooler to continue reading. At the same time even more storage was allocated to the

[1] Dijkstra, Aug. 1968.

[2] Dijkstra, May 1968.

> recursive program. After an interval in which the operator repeatedly forced further foolish storage allocations, the system finally ground to a complete halt, and a brief message explained that storage was exhausted and requested the operator to restart operations.
>
> So the answer was YES; the system did have a possibility of deadlock. But what interested me was that the restart message and the program that printed it were permanently resident in expensive core storage, so that it would be available even when the paging store and input/output utilities were inoperative. And secondly, that this was the very first time it had happened. I concluded that the THE operating system had been designed by a practical engineer of high genius. Having conducted the most fundamental and far-reaching research into deadlock and its avoidance, he nevertheless allocated scarce resources to ensure that if anything went wrong, it would be recognized and rectified. And finally, of course, nothing actually ever did go wrong, except as a demonstration to an inquisitive visitor.

David Gries remembered that Dijkstra's main contributions have been in programming methodology, and that he was one of the founders of IFIP Working Group 2.3 in 1970. On the other hand the remembrance indicated that Dijkstra is not infallible:

> WG 2.3 met in a log hotel overlooking Oslo during the week that man landed on the moon. During that summer week, Edsger, slightly short of breath while climbing a steep hill during an outing, said he did not believe that programming as a field of research would last another ten years-fifteen at the outside. Wad Turski says it is a pity that he did not challenge Edsger with a bet at the time, for he would have won. Turski was hesitant to bet because he had just lost a case of cognac: almost a decade earlier, at a New Year's party in Moscow, Turski bet a Russian scientist that man would not set foot on the moon before December 31, 1969, so Wad had just lost that bet by five months!

An unsubstantiated anecdote illustrates to what lengths people go to get the upper hand on Edsger:

> After Carel Scholten had built one of the early computers at the Mathematical Centre [Mathematisch Centrum, Amsterdam], Edsger claimed that nobody could write a shorter routine than his for some problem, and he offered a free meal to whoever could beat his routine (quite a bold bet for a Dutchman). He lost his bet, because Carel Scholten secretly added an instruction to the machine just so that he could write a shorter program! Thus, Edsger lost his one and only bet!

Dijkstra watchers, be they students of his lectures, or lecturers who have had him in the audience, are often perturbed by his lecturing and listening activities. Students are irritated by his habit of pausing between sentences to think about what he is to say next. Asked about it on one occasion he pointed out that English is not his native language and he picked up the habit early in his using the language. Doug McIlroy (Bell Telephone Laboratories) recalled the penury of a speaker who finds Dijkstra in his audience:

> As the speaker drones on, Edsger will become displeased at something, or begin thinking about something the speaker said. The body will rise, the sandals will come off, and the walking at the back of the room will begin. The unsuspecting new lecturer will continue blithely on. A more experienced lecturer will suspect and begin to worry. If he can contain himself, Edsger will wait until the end of the lecture, but sometimes he just has to interrupt. A snort will erupt, the nostrils will flare, the chin will elevate, and out will come an inspired, amazingly logical and eloquent, commentary. Both parties will emerge pleased, one for having vanquished stupidity, the other for having evoked the commentary and

> for the understanding they have gained. In the long run, this supreme effort of abrasion has polished the understanding of both.

McIlroy recalls only once that an eruption went supercritical. Unfortunately, the verbal outburst was saved for the end, and when it came, it lacked all divine inspiration: *"This stuff makes me sick!"* he thundered. Understanding was nonetheless polished, and two years later Dijkstra had taken up the topic himself.

David Gries (Cornell University) was one of the recipients of "on-line" coaching during a lecture:

> My own experience with lecturing before Edsger took place in Marktoberdorf. It rests on the fact that in some languages (notably Fortran), the equality symbol and the assignment symbol are the same, and many people say *"x equals e"* when they mean *"store the value of e in x,"* or *"x becomes e."*[1]
>
> I was lecturing along, when I said *"x equals e" meaning* an assignment of e to x. From the back of the room came a loud *"becomes," and* then a stunned silence. Finally, I gathered my wits and said, "Thank you, Edsger, for correcting me. If I make the same mistake again, stop me." Twenty minutes later, I made the same mistake, and again from the back of the room came *"becomes" "Thanks, I won't make the mistake again,"* I said, and to this day I haven't!

Without doubt Edsger Dijkstra, for all his technological contributions, epitomized by many to be the "GOTO" letter in the *Communications of the ACM,* is one of the "characters" of the field. He is difficult to predict. The titles of the two lectures he gave on accepting the ACM Turing Award and the SIGCSE Education Award are typical of this proclivity: "The Humble Programmer" and "On the Cruelty of Really Teaching Computer Science."

QUOTATIONS

"The question of whether computers can think is just like the question of whether submarines can swim." (Attrib.; posted on the CMU Board, December 1986)

I would require of a programming language that it should facilitate the work of the programmer as much as possible, especially in the most difficult aspects of his task, such as creating confidence in the correctness of his program. This is already difficult in the case of a specific program that must produce a finite set of results. But then the programmer only has the duty to show (afterwards) that if there were any flaws in his program they apparently didn't matter . . .' ("On the Design of Machine Independent Programming Languages," *Ann. Rev. in Auto. Prog., Vol. 3)*

"For the absence of a bibliography I offer neither explanation nor apology." *(A Discipline of Programming)*

"Program testing can be used to show the presence of bugs, but never to show their absence." *(Structured Programming, 1969 NATO* Conference)

---

[1] More precisely one should say "store the representation of the value *e* in the memory location associated with the name *x*." Ed.

“Suffering as I am from the sequential nature of human communication .....

BIBLIOGRAPHY

Biographical

Dijkstra, E.W., “The Humble Programmer,” Turing Award Lecture, *Comm. ACM*, Vol. 15, No. 10, Oct. 1972, pp. 859-866.

Dijkstra, Edsger W., “A Programmer's Early Memories,” in Metropolis, N., J. Howlett, and Gian-Carlo Rota, *A History of Computing in the Twentieth Century*, Academic Press, New York, 1980, pp. 563-573.

Lee, J.A.N., ed., “Frontiers of Computing: A Tribute to Edsger W. Dijkstra on the Occasion of his 60th Birthday,” *Ann. Hist. Comp*., Vol. 13, No. 1, 1991, pp. 91-96.

Significant Publications

Dijkstra, E.W., “Some Meditations on Advanced Programming,” *Proc. IFIP Congress*, North-Holland, Amsterdam, 1962, pp. 535-538.

Dijkstra, E.W., “Programming Considered as a Human Activity,” *Proc. IFIP Congress*, 1965, pp. 213-217.

Dijkstra, E.W., “Solution to a Problem in Concurrent Programming Control,” *Comm. ACM*, 1965, Vol. 8,1965, p. 569.

Dijkstra, E.W., “The Structure of the 'THE'-Multiprogramming System,” *ACM Symp. on Operating Systems, Comm. ACM*, Vol. 11, No. 5, May 1968, pp. 341-346.

Dijkstra, E.W., “GO TO Statement Considered Harmful,” letter to the editor, *Comm. ACM*, Vol. 11, No. 8, Aug. 1968, p. 538.

Dijkstra, E.W., *A Short Introduction to the Art of Computer Programming*, Technische Hogeschool, Eindhoven, 1971.

Dijkstra, E.W., “The Humble Programmer,” Turing Award Lecture, *Comm. ACM*, Vol. 15, No. 10, Oct. 1972, pp. 859-866.

Dijkstra, Edsger, *A Discipline of Programming*, Prentice-Hall, Englewood Cliffs, NJ., 1976.

Dijkstra, E.W., “On the Cruelty of Really Teaching Computer Science,” *Comm. ACM*, Vol. 32, No. 12, Dec. 1989, pp. 1398ff.

UPDATES

Edsger W. Dijkstra died August 6, 2002.

Dijkstra also received the following awards: an honorary doctorate from the Athens University of Economics and Business (2001) and the NEC Foundation Computer and Communications Prize (2002).

Portrait added. (MRW, 2012)