

Edsger W. Dijkstra

From Wikipedia, the free encyclopedia

Edsger Wybe Dijkstra (Dutch: [ˈɛtsxər ˈvibə ˈdɛikstra]; 11 May 1930 – 6 August 2002) was a Dutch computer scientist^[14] and an early pioneer in many research areas of computing science. A theoretical physicist by training, he worked as a programmer at the Mathematisch Centrum (Amsterdam) from 1952 to 1962. He was a professor of mathematics at the Eindhoven University of Technology (1962–1984) and a research fellow at the Burroughs Corporation (1973–1984). He held the Schlumberger Centennial Chair in Computer Sciences at the University of Texas at Austin from 1984 until 1999, and retired as Professor Emeritus in 1999.

One of the most influential members of computing science's founding generation, Dijkstra helped shape the new discipline from both an engineering and a theoretical perspective.^{[15][16]} Many of his papers are the source of new research areas. Several concepts and problems that are now standard in computer science were first identified by Dijkstra or bear names coined by him.^{[17][18]}

Computer programming in the 1950s to 1960s was not recognized as an academic discipline; Dijkstra, who had a background in mathematics and physics, was one of the moving forces behind the acceptance of computer programming as a scientific discipline.^{[19][20]} Dijkstra coined the phrase "structured programming" and during the 1970s this became the new programming orthodoxy.^{[21][22][23]} Dijkstra's ideas about structured programming helped lay the foundations for the birth and development of the professional discipline of software engineering, enabling programmers to organize and manage increasingly complex software projects.^{[24][25]}

The academic study of concurrent computing started in the 1960s, with Dijkstra (1965) credited with being the first paper in this field, identifying and solving the mutual exclusion problem.^[26] He was also one of the early pioneers of the research on principles of distributed computing. His foundational work on concurrency, semaphores, mutual exclusion (mutex), deadlock (deadly embrace), finding shortest paths in graphs, fault-tolerance, self-stabilization, among many other contributions comprises many of the pillars upon which the field of distributed computing is built. Shortly before his death in 2002, he received the ACM PODC Influential-Paper Award in distributed computing for his work on self-stabilization of program computation. This

Edsger Wybe Dijkstra



(2002)

Born	11 May 1930 <div>Rotterdam, Netherlands</div>
Died	6 August 2002 (aged 72) <div>Nuenen, Netherlands</div>
Fields	Computing science <div>Theoretical computer science</div>
Institutions	Mathematisch Centrum <div>Eindhoven University of Technology</div> <div>Burroughs Corporation</div> <div>The University of Texas at Austin</div>
Alma mater	Leiden University <div>(B.S., M.S.)</div> <div>University of Amsterdam</div> <div>(Ph.D.)</div>
Thesis	<i>Communication with an Automatic Computer</i> (1959)
Doctoral advisor	Adriaan van Wijngaarden
Doctoral students	Nico Habermann <div>Jan van de Snepscheut</div>
Known for	Dijkstra's algorithm (single-source shortest path problem) <div>DJP algorithm (minimum spanning tree problem)</div>

annual award was renamed the Dijkstra Prize (Edsger W. Dijkstra Prize in Distributed Computing) the following year, in his honor.^{[27][28][29]}

Contents

- 1 Biography
 - 1.1 Early years
 - 1.2 Mathematisch Centrum, Amsterdam
 - 1.3 Eindhoven University of Technology
 - 1.4 Burroughs Corporation
 - 1.5 The University of Texas at Austin
 - 1.6 Last years
- 2 Scientific contributions and impacts
 - 2.1 Algorithmic work
 - 2.2 Compiler construction and programming language research
 - 2.3 Programming paradigm and methodology
 - 2.4 Program design and development (software engineering research)
 - 2.5 Operating system research
 - 2.6 Concurrent computing and programming
 - 2.7 Distributed computing
 - 2.8 Formal specification and verification
 - 2.9 On the nature of computer science and computer programming
- 3 Personality and working style
- 4 EWDs
- 5 Personal life
- 6 Influence and recognition
- 7 Awards and honors
- 8 See also
- 9 Publications
- 10 Further reading
- 11 References
- 12 External links

Biography

Early years

Edsger W. Dijkstra was born in Rotterdam. His father was a chemist who was president of the Dutch Chemical Society; he taught chemistry at a secondary school and was later its superintendent. His mother was a mathematician, but never had a formal job.^{[30][31]}

Dijkstra had considered a career in law and had hoped to represent the Netherlands in the United Nations. However, after graduating from school in 1948, at his parents' suggestion he studied mathematics and physics and then theoretical physics at the University of Leiden.^[15]

First implementation of *ALGOL 60* (*Dijkstra-Zonneveld ALGOL 60* compiler for the Electrologica X1)

Structured analysis

Structured programming

Semaphore

Layered approach to operating system design

THE multiprogramming system

Concept of levels of abstraction^{[1][2]}

Concept of layered structure in software architecture (layered architecture)

Concept of cooperating sequential processes^[3]

Concept of program families^[4]

Multithreaded programming

Concurrent programming

Concurrent algorithms

Principles of distributed computing

Distributed algorithms

Synchronization primitive

Mutual exclusion

Critical section

Generalization of Dekker's algorithm

Tri-color marking algorithm

Call stack

Fault-tolerant systems

Self-stabilizing distributed systems

Resource starvation

Deadly embrace

Deadlock prevention algorithms

Shunting-yard algorithm

Banker's algorithm

Dining philosophers problem

Sleeping barber problem

Producer–consumer problem (bounded buffer problem)

Dutch national flag problem

Predicate transformer semantics

Guarded Command Language

In the early 1950s, electronic computers were a novelty. Dijkstra stumbled on his career quite by accident, and through his supervisor, Professor A. Haantjes, he met Adriaan van Wijngaarden, the director of the Computation Department at the Mathematical Center in Amsterdam, who offered Dijkstra a job; he officially became the Netherlands' first "programmer" in March 1952.^[15]

For some time Dijkstra remained committed to physics, working on it in Leiden three days out of each week. With increasing exposure to computing, however, his focus began to shift. As he recalled:^[32]

After having programmed for some three years, I had a discussion with A. van Wijngaarden, who was then my boss at the Mathematical Center in Amsterdam, a discussion for which I shall remain grateful to him as long as I live. The point was that I was supposed to study theoretical physics at the University of Leiden simultaneously, and as I found the two activities harder and harder to combine, I had to make up my mind, either to stop programming and become a real, respectable theoretical physicist, or to carry my study of physics to a formal completion only, with a minimum of effort, and to become....., yes what? A programmer? But was that a respectable profession? For after all, what was programming? Where was the sound body of knowledge that could support it as an intellectually respectable discipline? I remember quite vividly how I envied my hardware colleagues, who, when asked about their professional competence, could at least point out that they knew everything about vacuum tubes, amplifiers and the rest, whereas I felt that, when faced with that question, I would stand empty-handed. Full of misgivings I knocked on van Wijngaarden's office door, asking him whether I could "speak to him for a moment"; when I left his office a number of hours later, I was another person. For after having listened to my problems patiently, he agreed that up till that moment there was not much of a programming discipline, but then he went on to explain quietly that automatic computers were here to stay, that we were just at the beginning and could not I be one of the persons called to make programming a respectable discipline in the years to come? This was a turning point in my life and I completed my study of physics formally as quickly as I could.

	Weakest precondition calculus
	Unbounded nondeterminism
	Dijkstra-Scholten algorithm
	Smoothsort
	Separation of concerns
	Program verification
	Program derivation
	Software crisis ^[5]
	Software architecture ^[6]
Influences	Adriaan van Wijngaarden
Influenced	Tony Hoare ^[7]
	Niklaus Wirth ^{[8][9]}
	Per Brinch Hansen ^[10]
	Leslie Lamport ^{[11][12]}
	David Gries
	David Parnas
	Shlomi Dolev
	Alexander Stepanov ^[13]
Notable awards	Turing Award (1972)
	ACM Fellow (1994)
	Dijkstra Prize (2002)

— Edsger Dijkstra, The Humble
Programmer (EWD340), Communications
of the ACM

When Dijkstra married Maria (Ria) C. Debets in 1957, he was required as a part of the marriage rites to state his profession. He stated that he was a programmer, which was unacceptable to the authorities, there being no such profession at that time in The Netherlands.^{[33][34]}

In 1959 he received his PhD from the University of Amsterdam for a thesis entitled 'Communication with an Automatic Computer', devoted to a description of the assembly language designed for the first commercial computer developed in the Netherlands, the X1. His thesis supervisor was van Wijngaarden.^[17]

Mathematisch Centrum, Amsterdam

From 1952 until 1962 Dijkstra worked at the Mathematisch Centrum in Amsterdam,^[17] where he worked closely with Bram Jan Loopstra and Carel S. Scholten, who had been hired to build a computer. Their mode of interaction was disciplined: They would first decide upon the interface between the hardware and the software, by writing a programming manual. Then the hardware designers would have to be faithful to their part of the contract, while Dijkstra, the programmer, would write software for the nonexistent machine. Two of the lessons he learned from this experience were the importance of clear documentation, and that program debugging can be largely avoided through careful design.^[15] Dijkstra formulated and solved the shortest path problem for a demonstration at the official inauguration of the ARMAC computer in 1956, but — because of the absence of journals dedicated to automatic computing — did not publish the result until 1959.



At the Mathematical Center, Dijkstra and his colleague Jaap Zonneveld developed a compiler for the programming language ALGOL 60; it had a profound influence on his later thinking on programming as a scientific activity. He and Zonneveld had completed the implementation of the first ALGOL 60 compiler by August 1960, more than a year before a compiler was produced by another group.^[15]

Eindhoven University of Technology

In 1962 Dijkstra moved to Eindhoven, and later to Nuenen, in the south of the Netherlands, where he became a professor in the Mathematics Department at the Eindhoven University of Technology.^[17] The university did not have a separate computer science department and the culture of the mathematics department did not particularly suit him. Dijkstra tried to build a group of computer scientists who could collaborate on solving problems. This was an unusual model of research for the Mathematics Department.^[15] In the late 1960s he built the THE operating system (named for the university, then known as Technische Hogeschool Eindhoven), which has influenced the designs of subsequent operating systems.

Burroughs Corporation

Dijkstra joined Burroughs Corporation, a company known at that time for the production of computers based on an innovative hardware architecture, as its Research Fellow in August 1973. His duties consisted of visiting some of the company's research centers a few times a year and carrying on his own research, which he did in the smallest Burroughs research facility, namely, his study on the second floor of his house in Nuenen. In fact, Dijkstra was



The Eindhoven University of Technology, located in Eindhoven in the south of the Netherlands, where Dijkstra was a professor of mathematics from 1962 to 1984.

the only research fellow of Burroughs Corporation and worked for it from home, occasionally travelling to its branches in the United States. As a result, he reduced his appointment at the university to one day a week. That day, Tuesday, soon became known as the day of the famous 'Tuesday Afternoon Club', a seminar during which he discussed with his colleagues scientific articles, looking at all aspects – notation, organisation, presentation, language, content, etc. Shortly after he moved in 1984 to the University of Texas at Austin (USA), a new 'branch' of the Tuesday Afternoon Club emerged in Austin.^[17]

The Burroughs years saw him at his most prolific in output of research articles. He wrote nearly 500 documents in the EWD series (described below), most of them technical reports, for private circulation within a select group.^[15]

The University of Texas at Austin

Dijkstra accepted the Schlumberger Centennial Chair in the Computer Science Department at the University of Texas at Austin in 1984.

Last years

Dijkstra worked in Austin until his retirement in November 1999. To mark the occasion and to celebrate his forty-plus years of seminal contributions to computing science, the Department of Computer Sciences organized a symposium, which took place on his 70th birthday in May 2000.^[15]

Dijkstra and his wife returned from Austin to his original house in Nuenen (Netherlands) where he found that he had only months to live. He said that he wanted to retire in Austin, Texas, but to die in the Netherlands. Dijkstra died on 6 August 2002 after a long struggle with cancer.^[35] He and his wife Maria (Ria) Debets were survived by their three children: Marcus, Femke and the computer scientist Rutger M. Dijkstra.



The University of Texas at Austin, where Dijkstra held the Schlumberger Centennial Chair in Computer Sciences from 1984 until 1999.

Scientific contributions and impacts

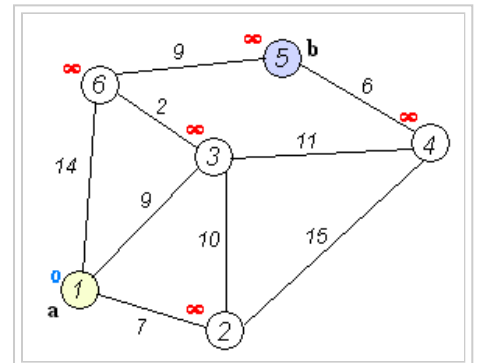
As an early theoretical pioneer in many research areas of computing science, Dijkstra helped shape the new discipline from both an engineering and an academic perspective. Many of his papers are the source of new research areas. Many concepts that are now standard in computer science were first identified by Dijkstra or bear names coined by him. Several important problems were also first formulated and solved by him. A 1994 survey of over a thousand professors of computer science was conducted to obtain a list of 38 most influential scholarly papers in the field, and Dijkstra is the author of five papers.^{[36][37][38]}

Algorithmic work

Dijkstra's algorithmic work (especially graph algorithms, concurrent algorithms, and distributed algorithms) plays an important role in many areas of computing science. According to Leslie Lamport (2002), Dijkstra "started the field of concurrent and distributed algorithms with his 1965 CACM paper "Solution of a Problem in Concurrent Programming Control", in which he first stated and solved the mutual exclusion problem." As Lamport explains, "that paper is probably why PODC exists (...). It remains to this day the most influential paper in the field. That it did not win a PODC Influential Paper Award reflects an artificial separation between concurrent and distributed algorithms—a separation that has never existed in Dijkstra's work."^[39]

In 1959 Dijkstra published in a 3-page article 'A note on two problems in connexion with graphs' the algorithm to find the shortest path in a graph between any two given nodes, now called Dijkstra's algorithm. Its impact over the next 40 years is summarised from the article of Mikkel Thorup, 'Undirected Single Source Shortest Paths with Positive Integer Weights in Linear Time' (1999): "Since 1959, all theoretical developments in SSSP

[Single-Source Shortest Paths] for general directed and undirected graphs have been based on Dijkstra's algorithm." Dijkstra's algorithm is used in SPF, Shortest Path First, which is used in the routing protocols OSPF and IS-IS. Various modifications to Dijkstra's algorithm have been proposed by many authors using heuristics to reduce the run time of shortest path search. One of the most used heuristic algorithms is the A* search algorithm, the main goal is to reduce the run time by reducing the search space. A* search algorithm (first described by Peter Hart, Nils Nilsson and Bertram Raphael of Stanford Research Institute in 1968)^[40] is an extension of Dijkstra's 1959 algorithm. Dijkstra thought about the shortest path problem when working at the Mathematical Center in Amsterdam in 1956 as a programmer to demonstrate capabilities of a new computer called ARMAC. His objective was to choose both a problem as well as an answer (that would be produced by computer) that non-computing people could understand. He designed the shortest path algorithm in about 20 minutes without aid of paper and pen and later implemented it for ARMAC for a slightly simplified transportation map of 64 cities in the Netherlands (so that 6 bits would suffice to represent the city in the algorithm).^{[41][42]}



Dijkstra's algorithm. It picks the unvisited vertex with the lowest-distance, calculates the distance through it to each unvisited neighbor, and updates the neighbor's distance if smaller. Mark visited (set to red) when done with neighbors.

A year later, he came across another problem from hardware engineers working on the institute's next computer: minimize the amount of wire needed to connect the pins on the back panel of the machine. As a solution, he re-discovered the algorithm known as Prim's minimal spanning tree algorithm. The Prim's algorithm was originally developed in 1930 by Czech mathematician Vojtěch Jarník^[43] and later independently rediscovered and republished by Robert C. Prim in 1957^[44] and Dijkstra in 1959.^[45] Therefore, it is also sometimes called the DJP algorithm.^[46]

In 1961 Dijkstra first described the shunting-yard algorithm, a method for parsing mathematical expressions specified in infix notation, in the Mathematisch Centrum report.^[47] It can be used to produce output in Reverse Polish notation (RPN) or as an abstract syntax tree (AST). The algorithm was named the "shunting yard" algorithm because its operation resembles that of a railroad shunting yard. The shunting-yard algorithm is commonly used to implement operator-precedence parsers.

In 1962 or 1963 Dijkstra proposed the semaphore mechanism for mutual exclusion algorithm for n processes (a generalization of Dekker's algorithm), which was probably the first published concurrent algorithm and which introduced a new area of algorithmic research. He also identified the deadlock problem and proposed the banker's algorithm that prevents deadlock.

In 1974 Dijkstra presented three self-stabilizing algorithms for mutual exclusion on a ring. Dijkstra's work is considered to be the first to introduce and demonstrate the self-stabilization concept.^[48]

In the mid-1970s Dijkstra (together with other authors) introduced two useful abstractions (mutator and collector) to the study of garbage collection. The mutator abstracts the process that performs the computation, including allocation of a new storage cell. The collector is the process that automatically reclaims garbage. Furthermore, this paper gives a formalization of "tri-color marking" that is basic to incremental garbage collection.^{[49][50]}

In the early 1980s Dijkstra and Carel S. Scholten proposed the Dijkstra–Scholten algorithm for detecting termination in distributed systems.

In 1981 Dijkstra developed smoothsort, a comparison-based sorting algorithm and a variation of heapsort.

Compiler construction and programming language research

Dijkstra was known to be a fan of ALGOL 60, and worked on the team that implemented the first compiler for that language. He was closely involved in the ALGOL 60 development, realisation and popularisation. As discussed by Peter Naur in the article 'The European side of the last phase of the development of ALGOL 60', in the *Proceedings of the First ACM SIGPLAN Conference on History of Programming Languages*, January 1978, Dijkstra took part in the period 1958–1959 in a number of meetings that culminated in the publication of the report defining the ALGOL 60 language. Dijkstra's name does not appear in the list of 13 authors of the final report. Apparently, he eventually left the committee because he could not agree with the majority opinions. Still, while at the Mathematisch Centrum (Amsterdam), he wrote jointly with Jaap Zonneveld the first ALGOL 60 compiler. Dijkstra and Zonneveld, who collaborated on the compiler, agreed not to shave until the project was completed; while Zonneveld shaved shortly thereafter, Dijkstra kept his beard for the rest of his life.^[51]



By August 1960, Dijkstra and his colleague Zonneveld had completed the implementation of the first ALGOL 60 compiler for the Electrologica X1 computer.

ALGOL was the result of a collaboration of American and European committees. ALGOL 60 (short for ALGOrithmic Language 1960) is a member of the ALGOL family of computer programming languages. It followed on from ALGOL 58 and inspired many languages that followed it. It gave rise to many other programming languages, including BCPL, B, Pascal, Simula and C.^[52] Algol 60 was a sophisticatedly designed computer language and it provided a large number of hitherto unknown implementation challenges. As Bjarne Stroustrup notes, "one problem with Algol60 was that no one knew how to implement it."^[53] A major new challenge in Algol 60 implementation was the run-time allocation and management of data. In 1960 Dijkstra and Zonneveld showed how recursive procedures could be executed using a run-time stack of activation records, and how to efficiently access identifiers from statically enclosing scopes using the so-called 'display'.^[54] The ALGOL 60 compiler was one of the first to support recursion^[55] employing a novel method to do so. Dijkstra's short book *Primer of Algol 60 Programming*, originally published in 1962, was the standard reference for the language for several years.

Programming paradigm and methodology

Computer programming in the 1950s to 1960s was not recognized as an academic discipline and unlike mature sciences there were no theoretical concepts or coding systems. Programming as a professional activity was poorly understood in those years.

In the late 1960s computer programming was in state of crisis. Software crisis is a term used in the early days of computing science for the difficulty of writing useful and efficient computer programs in the required time. The software crisis was due to the rapid increases in computer power and the complexity of the problems that could be tackled. With the increase in the complexity of the software, many software problems arose because existing methods were neither sufficient nor up to the mark. The term "software crisis" was coined by some attendees at the first NATO Software Engineering Conference in 1968 at Garmisch, Germany.^{[5][56][57]} His 1972 ACM Turing Award Lecture makes reference to this same problem: "The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem."^[32]

While Dijkstra had programmed extensively in machine code in the 1950s, he came to the conclusion that in high-level languages frequent use of the GOTO statement was usually symptomatic of poor structure. In 1968 he wrote a private paper "A Case against the GO TO Statement",^[58] which was then published as a letter in CACM.^[59] Editor Niklaus Wirth gave this letter the heading "Go To Statement Considered Harmful", which introduced the phrase "considered harmful" into computing.

Dijkstra argued that the programming statement GOTO, found in many high-level programming languages, is a major source of errors, and should therefore be eliminated. This letter caused a huge debate in the programming community. Some went to the length of equating good programming with the elimination of GOTO. Dijkstra refused to mention the debate, or even the GOTO statement, in his article "Notes on Structured Programming". The debate has long since died down; programming languages provide alternatives to the GOTO, few programmers today use it liberally, and most never use it at all.^[15]

Dijkstra's thesis was that departures from linear control flow were clearer if allowed only in disciplined higher-level structures such as the if-then-else statement and the while loop. This methodology was developed into structured programming movement, the title of his 1972 book, coauthored with C.A.R. Hoare and Ole-Johan Dahl. Considered by many as the first significant movement in history of computer programming, structured programming became the new programming orthodoxy during the 1970s.^{[60][61][62]} Bertrand Meyer remarked that, "The revolution in views of programming started by Dijkstra's iconoclasm led to a movement known as structured programming, which advocated a systematic, rational approach to program construction. Structured programming is the basis for all that has been done since in programming methodology, including object-oriented programming."^[63]

Structured programming is often regarded as "goto-less programming". But as Bertrand Meyer notes, "As the first book on the topic [*Structured Programming* by Dijkstra, Dahl, and Hoare] shows, structured programming is about much more than control structures and the goto. Its principal message is that programming should be considered a scientific discipline based on mathematical rigor."^[64] As a programming paradigm, structured programming – especially in the 1970s and 1980s – significantly influenced the birth of many modern programming languages such as Pascal, C, Modula-2, and Ada.^[65] The Fortran 77 version which incorporates the concepts of structured programming, was released in 1978. The C++ language was a considerably extended and enhanced version of the popular structured programming language C (see also: list of C-based programming languages). Since C++ was developed from a more traditional structured language, it is a 'hybrid language', rather than a pure object-oriented programming language.^[66]

Program design and development (software engineering research)

Dijkstra's ideas about programming methodology (especially the structured programming movement) helped lay the foundations for the birth and development of the professional discipline of software engineering (in particular the software design and development), enabling programmers to organize and manage increasingly complex software projects.^{[67][68]} In the late 1960s Dijkstra discussed the concept of program families. And in the mid 1970s David Parnas and others clarified the idea and showed how to apply it in software engineering principles.

The rise of the structured programming movement led to many other *structured* approaches applied to software design. The techniques of structured analysis and structured design are outgrowths of structured programming concepts and techniques, and of the early ideas about modular design. Principles of modularity were strengthened by Larry Constantine's concepts of coupling (to be minimized between modules) and cohesion (to be maximized within modules), by David Parnas's techniques of information hiding, and by abstract data types.^[69] A number of tools and methods employing structured concepts were developed, such as Structured Design, Jackson's Structured Programming, Ross' Structured Analysis and Design Technique (SADT), Yourdon's Structured Method, Structured Systems Analysis and Design Method (SSADM), and James Martin's Information Engineering. The field of software metrics is often considered as a direct influence of the structured programming movement on software engineering in the 1970s.

Separation of concerns (SoC), one of the basic principles in software engineering, is a design principle for separating a computer program into distinct sections, such that each section addresses a separate concern. The term *separation of concerns* was coined by Dijkstra in his 1974 paper "On the role of scientific thought".^[70]

Operating system research

In the 1960s Dijkstra and his colleagues in Eindhoven designed and implemented THE (standing for 'Technische Hogeschool Eindhoven') operating system, which was organised into clearly identified layers.^{[71][72]} His 1968 article on this subject provided the foundation for subsequent designs of the operating systems.

Dijkstra organized the design of the system in layers in order to reduce the overall complexity of the software. Though the term 'architecture' had not yet been used to describe software design, this was certainly considered the first glimpse of software architecture.^[73] It introduced a number of design principles which have become part of the working vocabulary of every professional programmer: levels of abstraction, programming in layers, the semaphore, and cooperating sequential processes. His original paper on the THE operating system was reprinted in the 25th Anniversary issue of Communications of the ACM, in January 1983. By way of introduction, the Editor-in-Chief says, "This project initiated a long line of research in multilevel systems architecture — a line that continues to the present day because hierarchical modularity is a powerful approach to organizing large systems."^[15]

Concurrent computing and programming

In a one-page paper from 1965 Dijkstra introduced the 'mutual exclusion problem' for n processes and discussed a solution to it. It was probably the first published concurrent algorithm.^{[12][17]} The notion, standard by now, of a 'critical section' was also introduced in this paper. Per Brinch Hansen, a pioneer in the field of concurrent computing, considers Dijkstra's *Cooperating Sequential Processes* (1965) to be the first classic paper in concurrent programming. As Brinch Hansen notes, 'Dijkstra lays the conceptual foundation for abstract concurrent programming' with that paper.^[75]

In 1968 Dijkstra published his seminal paper 'Cooperating sequential processes', a 70-page essay that originated the field of concurrent programming.^[76] He discussed in it the notion of mutual exclusion (mutex) and the criteria a satisfactory solution should satisfy. He also redressed the historical perspective left out of his 1965 paper by including the first known correct solution to the mutual exclusion problem, for two processes, due to Theodorus Dekker. Dijkstra subsequently generalized Dekker's solution to n processes.^{[77][78]} Further, he proposed the first synchronisation mechanism for concurrent processes,^[79] the semaphore with its two operations, P and V. He also identified the 'deadlock problem' (called there 'the problem of the deadly embrace')^[80] and proposed an elegant 'Banker's algorithm' that prevents deadlock. The deadlock detection and prevention became perennial research problems in the field of concurrent programming.

The dining philosophers problem is an example problem often used in concurrent algorithm design to illustrate synchronization issues and techniques for resolving them. It was originally formulated in 1965 by Dijkstra as a student exam exercise, presented in terms of computers competing for access to tape drive peripherals. Soon after, Tony Hoare gave the problem its present formulation.^[81] The sleeping barber problem is also attributed to Dijkstra.

Distributed computing

Dijkstra was one of the very early pioneers of the research on principles of distributed computing.^{[82][83]} As the citation for the Dijkstra Prize recognizes, "no other individual has had a larger influence on research in principles of distributed computing." Some of his papers are even considered to be those that established the field. Dijkstra's 1965 paper, *Solution of a Problem in Concurrent Programming Control* was the first to present



A semaphore (Dutch: *seinpaal*, the term used in Dijkstra's original description^[74]). In the early 1960s Dijkstra proposed the first synchronisation mechanism for concurrent processes, the semaphore with its two operations, P and V.

the correct solution to the mutual exclusion problem. Leslie Lamport writes that this work "is probably why PODC exists" and it "started the field of concurrent and distributed algorithms".^[84]

In particular, his paper "Self-stabilizing Systems in Spite of Distributed Control" (1974) started the sub-field of self-stabilization. It is also considered as the first scientific examination of fault-tolerant systems.^[12] Dijkstra's paper was not widely noticed until Leslie Lamport's invited talk at the ACM Symposium on Principles of Distributed Computing (PODC) in 1983. In his report on Dijkstra's work on self-stabilizing distributed systems, Lamport regard it to be 'a milestone in work on fault tolerance' and 'a very fertile field for research'.^[85]

Formal specification and verification

From the 1970s, Dijkstra's chief interest was formal verification. In 1976 Dijkstra published a seminal book, *A Discipline of Programming*, which put forward his method of systematic development of programs together with their correctness proofs. In his exposition he used his 'Guarded Command Language'. The language, with its reliance on non-determinism, the adopted weakest precondition semantics and the proposed development method has had a considerable impact on the field to this day. The refinement calculus, originally proposed by Ralph-Johan Back^[86] and developed by Carroll Morgan,^[87] is an extension of Dijkstra's weakest precondition calculus, where program statements are modeled as predicate transformers.^[88]

In 1984, to add further support to this approach to programming, he published jointly with Wim Feijen an introductory textbook for first-year students of computer science. The book, first published in Dutch, was entitled *Een methode van programmeren*. The English edition appeared in 1988 as *A Method of Programming*.

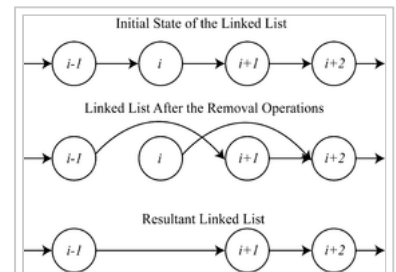
On the nature of computer science and computer programming

Many of his opinions on computer science and programming have become widespread. For example, the programming phrase "two or more, use a for" (a rule of thumb when to use a loop) is sometimes attributed to him.^[89]

He was the first to make the claim that programming is so inherently complex that, in order to manage it successfully, programmers need to harness every trick and abstraction possible.

Dijkstra was one of the most famous opponents of the engineering view of computing science. Like Peter Naur and Kristen Nygaard, Dijkstra disliked the very term 'computer science'. Computer science, as Dijkstra pointed out, deserves a better name. He suggests it can be called 'computing science'. Instead of the computer, or computing technology, Dijkstra wanted to emphasize the abstract mechanisms that computing science uses to master complexity. When expressing the abstract nature of computing science, he wrote,

A confusion of even longer standing came from the fact that the unprepared included the electronic engineers that were supposed to design, build and maintain the machines. The job was actually beyond the electronic technology of the day, and, as a result, the question of how to get and keep the physical equipment more or less in working condition became in the early days the all-overriding concern. As a result, the topic became – primarily in the USA – prematurely known as



A simple example of two processes modifying a linked list at the same time causing a conflict. The requirement of mutual exclusion was first identified and solved by Dijkstra in his seminal 1965 paper titled *Solution of a problem in concurrent programming control*, and is credited as the first topic in the study of concurrent algorithms.



Illustration of the dining philosophers problem

‘computer science’ – which, actually, is like referring to surgery as ‘knife science’ – and it was firmly implanted in people's minds that computing science is about machines and their peripheral equipment. Quod non [Latin: "Which is not true"]. We now know that electronic technology has no more to contribute to computing than the physical equipment. We now know that programmable computer is no more and no less than an extremely handy device for realizing any conceivable mechanism without changing a single wire, and that the core challenge for computing science is hence a conceptual one, viz., what (abstract) mechanisms we can conceive without getting lost in the complexities of our own making.^[90]

In *The Humble Programmer* (1972), Dijkstra wrote: "We must not forget that it is not our [computing scientists'] business to make programs, it is our business to design classes of computations that will display a desired behaviour."

Dijkstra also opposed the inclusion of software engineering under the umbrella of academic computer science. He wrote that, "As economics is known as "The Miserable Science", software engineering should be known as "The Doomed Discipline", doomed because it cannot even approach its goal since its goal is self-contradictory." And "software engineering has accepted as its charter "How to program if you cannot."."^[91]

Personality and working style



Dijkstra at the blackboard during a conference at ETH Zurich in 1994

In the world of computing science, Dijkstra is well known as a "character". In the preface of his book *A Discipline of Programming* (1976) he stated the following: "For the absence of a bibliography I offer neither explanation nor apology." In fact, most of his articles and books have no references at all.^[17] This approach to references was deplored by some researchers. But Dijkstra chose this way of working to preserve his self-reliance.

As a university professor for much of his life, Dijkstra saw teaching not just as a required activity but as a serious research endeavor.^[15] His approach to teaching was unconventional.^[92] His lecturing style has been described as idiosyncratic. When lecturing, the long pauses

between sentences have often been attributed to the fact that English is not Dijkstra's first language. However the pauses also served as a way for him to think on his feet and he was regarded as a quick and deep thinker while engaged in the act of lecturing. His courses for students in Austin had little to do with computer science but they dealt with the presentation of mathematical proofs.^[17] At the beginning of each semester he would take a photo of each of the students, in order to memorize their names. He never followed a textbook, with the possible exception of his own while it was under preparation. When lecturing, he would write proofs in chalk on a blackboard rather than using overhead foils. He invited the students to suggest ideas, which he then explored, or refused to explore because they violated some of his tenets. He assigned challenging homework problems, and would study his students' solutions thoroughly. He conducted his final examinations orally, over a whole week. Each student was examined in Dijkstra's office or home, and an exam lasted several hours.^[15]

He was also highly original in his way of assessing people's capacity for a job. When Vladimir Lifschitz came to Austin in 1990 for a job interview, Dijkstra gave him a puzzle. Vladimir solved it and has been working in Austin since then.^[17]

Despite having invented much of the technology of software, Dijkstra eschewed the use of computers in his own work for many decades. Even after he succumbed to his UT colleagues' encouragement and acquired a Macintosh computer, he used it only for e-mail and for browsing the World Wide Web.^[93] Dijkstra never wrote

his articles using a computer. He preferred to rely on his typewriter and later on his Montblanc pen.^[17] Dijkstra's favorite writing instrument was the Montblanc Meisterstück fountain pen. He repeatedly tried other pens, but none ever displaced the Montblanc.

He had no use for word processors, believing that one should be able to write a letter or article without rough drafts, rewriting, or any significant editing. He would work it all out in his head before putting pen to paper, and once mentioned that when he was a physics student he would solve his homework problems in his head while walking the streets of Leiden.^[15] Most of Dijkstra's publications were written by him alone. He never had a secretary and took care of all his correspondence alone.^[17] When colleagues prepared a Festschrift for his sixtieth birthday, published by Springer-Verlag, he took the trouble to thank each of the 61 contributors separately, in a hand-written letter.^[17]

Throughout Dijkstra's career, his work was characterized by elegance and economy.^[17] A prolific writer (especially as an essayist), Dijkstra authored more than 1,300 papers, many written by hand in his precise script. They were essays and parables; fairy tales and warnings; comprehensive explanation and pedagogical pretext. Most were about mathematics and computer science; others were trip reports that are more revealing about their author than about the people and places visited. It was his habit to copy each paper and circulate it to a small group of colleagues who would copy and forward the papers to another limited group of scientists.^[94] His love affair with simplicity came at an early age and under his mother's guidance. He once said he had asked his mother whether trigonometry was a difficult topic. She replied that he must learn all the formulas and that furthermore if he required more than five lines to prove something, he was on the wrong track.^[95]

Dijkstra was famous for his wit, eloquence, and way with words, such as in his remark, "The question of whether Machines Can Think (...) is about as relevant as the question of whether Submarines Can Swim."; his advice to a promising researcher, who asked how to select a topic for research, "Do only what only you can do".^[15] Dijkstra was also known for his vocal criticism. As an outspoken and critical visionary, he strongly opposed the teaching of BASIC.^[96]

In many of his more humorous essays, Dijkstra described a fictional company of which he served as chairman. The company was called Mathematics, Inc., a company that he imagined having commercialized the production of mathematical theorems in the same way that software companies had commercialized the production of computer programs. He invented a number of activities and challenges of Mathematics Inc. and documented them in several papers in the EWD series. The imaginary company had produced a proof of the Riemann Hypothesis but then had great difficulties collecting royalties from mathematicians who had proved results assuming the Riemann Hypothesis. The proof itself was a trade secret.^[97] Many of the company's proofs were rushed out the door and then much of the company's effort had to be spent on maintenance.^[98] A more successful effort was the Standard Proof for Pythagoras' Theorem, that replaced the more than 100 incompatible existing proofs.^[99] Dijkstra described Mathematics Inc. as "the most exciting and most miserable business ever conceived".^[97] EWD 443 (1974) describes his fictional company as having over 75 percent of the world's market share.^{[100][101]}

EWDs

Dijkstra was well known for his habit of carefully composing manuscripts with his fountain pen. The manuscripts are called EWDs, since Dijkstra numbered them with *EWD*, his initials, as a prefix. According to Dijkstra himself, the EWDs started when he moved from the Mathematical Centre in Amsterdam to the Eindhoven University of Technology (then Technische Hogeschool Eindhoven). After going to Eindhoven, Dijkstra experienced a writer's block for more than a year. Dijkstra distributed photocopies of a new EWD among his colleagues. Many recipients photocopied and forwarded their copies, so the EWDs spread throughout the international computer science community. The topics were computer science and mathematics, and included trip reports, letters, and speeches. These short articles span a period of 40 years. Almost all EWDs

appearing after 1972 were hand-written. They are rarely longer than 15 pages and are consecutively numbered. The last one, No. 1318, is from 14 April 2002. Within computer science they are known as the EWD reports, or, simply the EWDs. More than 1300 EWDs have been scanned, with a growing number transcribed to facilitate search, and are available online at the Dijkstra archive of the University of Texas.^[102]

Personal life

Dijkstra's self-confidence went together with a remarkably modest lifestyle, to the point of being spartan.^[17] His and his wife's house in Nuenen was simple, small and unassuming. He did not own a TV, a VCR or a mobile telephone, and did not go to the movies.^[17] In contrast, he played the piano well and, while in Austin, liked to go to concerts. An enthusiastic listener of classical music, Dijkstra's favorite composer was Mozart.^[15]

Influence and recognition

In 1972 the Association for Computing Machinery (ACM) acknowledged Dijkstra's seminal contributions to the field by awarding him the distinguished Turing Award. The citation for the award reads:^[104]

Edsger Dijkstra was a principal contributor in the late 1950's to the development of the ALGOL, a high level programming language which has become a model of clarity and mathematical rigor. He is one of the principal exponents of the science and art of programming languages in general, and has greatly contributed to our understanding of their structure, representation, and implementation. His fifteen years of publications extend from theoretical articles on graph theory to basic manuals, expository texts, and philosophical contemplations in the field of programming languages.

“ The difference between a computer programmer and a computer scientist is a job-title thing. Edsger Dijkstra wants proudly to be called a "computer programmer," although he hasn't touched a computer now for some years. (...) His great strength is that he is uncompromising. It would make him physically ill to think of programming in C++. ”

The introduction given at the awards ceremony is a tribute to Dijkstra:^[104]

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 influence on programming is more pervasive than any glossary can possibly indicate. The precious gift that this Turing Award acknowledges is 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. (...) We have come to value good programs in much the same way as we value good literature. And at the center of this movement, creating and reflecting patterns no less beautiful than useful, stands E.W. Dijkstra.

— Donald Knuth (1996), an interview with Donald Knuth by Jack Wöehr of *Dr. Dobbs's Journal*.^[103]

In the words of Sir Tony Hoare, FRS, delivered by him at Dijkstra's funeral:^[15]

Edsger is widely recognized as a man who has thought deeply about many deep questions; and among the deepest questions is that of traditional moral philosophy: How is it that a person should live their life? Edsger found his answer to this question early in his life: He decided he would live as an academic scientist, conducting research into a new branch of science, the science of computing. He would lay the foundations that would establish computing as a rigorous scientific discipline; and in his research and in his teaching and in his writing, he would pursue perfection to the exclusion of all other concerns. From these commitments he never deviated, and that is how he has made to his chosen subject of study the greatest contribution that any one person could make in any one lifetime.

In March 2003, the following email was sent to the distributed computing community:^[105]

This is to announce that the award formerly known as the "PODC Influential-Paper Award" has been renamed the "Edsger W. Dijkstra Prize in Distributed Computing" after the late Edsger W. Dijkstra, a pioneer in the area of distributed computing. His foundational work on concurrency primitives (such as the semaphore), concurrency problems (such as mutual exclusion and deadlock), reasoning about concurrent systems, and self-stabilization comprises one of the most important supports upon which the field of distributed computing is built. No other individual has had a larger influence on research in principles of distributed computing.

Former ACM President Peter J. Denning wrote about Dijkstra:^[106]

Edsger Dijkstra, one of the giants of our field and a passionate believer in the mathematical view of programs and programming (...) Over the previous quarter-century, he had formulated many of the great intellectual challenges of the field as programming—the goto statement, structured programming, concurrent processes, semaphores, deadlocks, recursive programming in Algol, and deriving correct programs.

Awards and honors

Among Dijkstra's awards and honors are:^[93]

- Member of the Royal Netherlands Academy of Arts and Sciences (1971)^[107]
- Distinguished Fellow of the British Computer Society (1971)
- The Association for Computing Machinery's A.M. Turing Award (1972)^[108]
- Harry H. Goode Memorial Award from the IEEE Computer Society (1974).^[109]
- Foreign Honorary Member of the American Academy of Arts and Sciences (1975)
- Doctor of Science Honoris Causa from the Queen's University Belfast (1976)
- Computer Pioneer Charter Recipient from the IEEE Computer Society (1982)
- ACM/SIGCSE Award for Outstanding Contributions to Computer Science Education (1989)
- Fellow of the Association for Computing Machinery (1994)^[110]
- Honorary doctorate from the Athens University of Economics & Business, Greece (2001).

The Distinguished Fellowship of the British Computer Society (BCS) is awarded under bylaw 7 of the BCS's Royal Charter. The award was first approved in 1969 and the first election was made in 1971 to Dijkstra.^[111]

On the occasion of Dijkstra's 60th birthday in 1990, The Department of Computer Science (UTCS) at the University of Texas at Austin organized a two-day seminar in his honor. Speakers came from all over the United States and Europe, and a group of computer scientists contributed research articles which were edited into a book.^{[15][112]}

In 2002, the C&C Foundation of Japan recognized Dijkstra "for his pioneering contributions to the establishment of the scientific basis for computer software through creative research in basic software theory, algorithm theory, structured programming, and semaphores." Dijkstra was alive to receive notice of the award, but it was accepted by his family in an award ceremony after his death.

Shortly before his death in 2002, Dijkstra received the ACM PODC Influential-Paper Award in distributed computing for his work on self-stabilization of program computation. This annual award was renamed the Dijkstra Prize (Edsger W. Dijkstra Prize in Distributed Computing) the following year, in his honor.

The Dijkstra Award for Outstanding Academic Achievement in Computer Science (Loyola University Chicago, Department of Computer Science) is named for Edger W. Dijkstra. Beginning in 2005, this award recognizes the top academic performance by a graduating computer science major. Selection is based on GPA in all major courses and election by department faculty.^[113]

The Department of Computer Science (UTCS) at the University of Texas at Austin hosted the inaugural Edsger W. Dijkstra Memorial Lecture on October 12, 2010. Tony Hoare, Emeritus Professor at Oxford and Principal Researcher at Microsoft Research, was the speaker for the event. This lecture series was made possible by a generous grant from Schlumberger to honor the memory of Dijkstra.

See also

- Dijkstra's algorithm
- Dining philosophers problem
- Guarded Command Language
- Predicate transformer semantics
- Weakest precondition calculus
- Semaphore
- Smoothsort
- *Go To Statement Considered Harmful*
- *On the Cruelty of Really Teaching Computer Science*
- List of pioneers in computer science
- List of important publications in computer science
- List of important publications in theoretical computer science
- List of important publications in concurrent, parallel, and distributed computing

Publications

Books:

- Dijkstra, Edsger W.: *A Primer of ALGOL 60 Programming: Together with Report on the Algorithmic Language ALGOL 60*. (New York: Academic Press, 1962)
- Dijkstra, Edsger W.; Dahl, Ole-Johan; Hoare, C. A. R. (1972). *Structured Programming*. Academic Press. ISBN 0-12-200550-3.
- Dijkstra, Edsger W.: *A Discipline of Programming*. (Prentice Hall, 1976, 217pp)
- Dijkstra, Edsger W.: *Selected Writings on Computing: A Personal Perspective (Monographs in Computer Science)*. (Springer, 1982, 362pp)
- Dijkstra, Edsger W.; Feijen, W. H. J.; Sterringa, Joke: *A Method of Programming*. (Addison-Wesley, 1988, 198pp)
- Dijkstra, Edsger W.; Scholten, Carel S.: *Predicate Calculus and Program Semantics (Texts and Monographs in Computer Science)*. New York: Springer-Verlag, 1990, 220pp

Selected articles:

- Dijkstra, Edsger W. (1959). "A Note on Two Problems in Connexion with Graphs". *Numerische Mathematik*. **23** (3): 269–271. doi:10.1007/BF01386390.
- Dijkstra, Edsger W. (1962). "Some Meditations on Advanced Programming," Proc. IFIP Congress, North-Holland, Amsterdam, pp. 535–538.
- Dijkstra, Edsger W. (1965). *Cooperating Sequential Processes* (Technische Hogeschool Eindhoven). Reprinted in F. Genuys (ed.), *Programming Languages*, Academic Press, Orlando, Florida, 1968, pp. 43–112
- Dijkstra, Edsger W. (1965). "Solution of a Problem in Concurrent Programming Control". *Communications of the ACM* 8 (9): 569.
- Dijkstra, Edsger W. (1965). "Programming Considered as a Human Activity," Proc. IFIP Congress, pp. 213–217.
- Dijkstra, Edsger W. (1968). *Letters to the editor: Go To Statement Considered Harmful*. *Commun. ACM* 11(3): 147-148
- Dijkstra, Edsger W. (1968). *A Constructive Approach to the Problem of Program Correctness*. *BIT Numerical Mathematics* 8(1968): pp. 174–186.
- Dijkstra, Edsger W. (1968). "The Structure of the 'THE'-Multiprogramming System," *ACM Symp. on Operating Systems*, *Comm. ACM*, Vol. 11, No. 5, May 1968, pp. 341–346.
- Dijkstra, Edsger W. (1971). *A Short Introduction to the Art of Computer Programming*. (Technische Hogeschool, Eindhoven)
- Dijkstra, Edsger W. (1971). *Hierarchical Ordering of Sequential Processes*. *Acta Inf.* 1: 115-138
- Dijkstra, Edsger W. (1972). *The Humble Programmer*. *Communications of the ACM* 15(10): pp. 859-866.
- Dijkstra, Edsger W. (1972). *Notes on Structured Programming*, in *Structured Programming*, by O.-J. Dahl, E. W. Dijkstra, and C. A. R. Hoare, New York: Academic Press
- Dijkstra, Edsger W. (1974). *Programming as a Discipline of Mathematical Nature*. *American Mathematical Monthly* 81(JuneJuly): pp. 608-612.
- Dijkstra, Edsger W. (1974). *On the role of scientific thought* (EWD447). (E.W. Dijkstra Archive, Center for American History, University of Texas at Austin)
- Dijkstra, Edsger W. (1974). *Self-stabilizing Systems in Spite of Distributed Control*. *Commun. ACM* 17(11): 643-644
- Dijkstra, Edsger W. (1975). *How do we tell truths that might hurt?*. In Dijkstra, Edsger W. (1982): "Selected Writings on Computing: A Personal Perspective": pp. 129-131.
- Dijkstra, Edsger W. (1975). *Craftsman or Scientist*. *ACM Pacific* 1975: 217-223
- Dijkstra, Edsger W. (1975). *On the teaching of programming, i. e. on the teaching of thinking*. *Language Hierarchies and Interfaces* 1975: 1-10
- Dijkstra, Edsger W. (1977). *Programming: From Craft to Scientific Discipline*. *International Computing Symposium* 1977: 23-30
- Dijkstra, Edsger W. (1978). *On the Interplay between Mathematics and Programming*. *Program Construction* 1978: 35-46
- Dijkstra, Edsger W. (1975). *Correctness Concerns And, Among Other Things, Why They Are Resented*. (ACM) *Proceedings of the international conference on Reliable software*. April 21–23, 1975, Los Angeles, California, USA: pp. 546-550.
- Dijkstra, Edsger W. (1975). *Guarded Commands, Nondeterminacy and Formal Derivation of Programs*. *Commun. ACM* 18(8): 453-457
- Dijkstra, Edsger W. (1978). *Finding the Correctness Proof of a Concurrent Program*. *Program Construction* 1978: 24-34
- Dijkstra, Edsger W. (1984). *The threats to computing science* (EWD898). (E.W. Dijkstra Archive, Center for American History, University of Texas at Austin)
- Dijkstra, Edsger W. (1986). *On a Cultural Gap*. *The Mathematical Intelligencer* 8(1): pp. 48-52.
- Dijkstra, Edsger W. (1987). *Mathematicians and Computing Scientists: The Cultural Gap*. *Abacus* 4(4): pp. 26-31.
- Dijkstra, Edsger W. (1989). *On the Cruelty of Really Teaching Computer Science*. *Communications of the ACM* 32(12): pp.1398-1404.
- Dijkstra, Edsger W. (1999). *Computing Science: Achievements and Challenges*. *ACM SIGAPP Applied Computing Review* 7(2): pp. 2-9.
- Dijkstra, Edsger W. (2001). *The End of Computing Science?*. *Communications of the ACM* 44(3): pp. 92.

- Dijkstra, Edsger W. (2001). "What led to *Notes on Structured Programming*". (E.W. Dijkstra Archive, Center for American History, University of Texas at Austin)

Further reading

- Apt, Krzysztof R. (2002). "Edsger Wybe Dijkstra (1930-2002): A Portrait of a Genius". *Formal Aspects of Computing*, Volume 14, Issue 2, pp. 92–98
- Brinch Hansen, Per: *The Origin of Concurrent Programming: From Semaphores to Remote Procedure Calls*. (Springer, 2002, 534pp)
- Ben-Ari, Mordechai: *Principles of Concurrent and Distributed Programming*, 2nd edition. (Pearson Education Limited, 2006, 384 pp)
- Broy, Manfred; Denert, Ernst (eds.): *Software Pioneers: Contributions to Software Engineering*. (Springer, 2002, 728pp)
- Daylight, Edgar G.: *The Dawn of Software Engineering: from Turing to Dijkstra*. (Lonely Scholar, 2012, ISBN 9789491386022)
- Daylight, Edgar G.: *Dijkstra's Rallying Cry for Generalization: The Advent of the Recursive Procedure, Late 1950s–Early 1960s*. (The Computer Journal (2011) 54 (11): 1756-1772.[doi: 10.1093/comjnl/bxr002])
- Dolev, Shlomi: *Self-stabilization*. (MIT Press, 2000, 207pp)
- Feijen, W.; van Gasteren, A.J.M.; Gries, D.; Misra, J. (eds.): *Beauty Is Our Business: A Birthday Salute to Edsger W. Dijkstra*. (Springer, 1990, 455pp)
- Laplante, Phillip (ed.): *Great Papers in Computer Science*. (Minneapolis-St. Paul: West Publishing Company, 1996, 600pp)
- Lee, J.A.N. (1991). *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.)
- O'Regan, Gerard: *Giants of Computing: A Compendium of Select, Pivotal Pioneers*. (Springer, 2013, 306pp)
- Payette, Sandy (2014). *Hopper and Dijkstra: Crisis, Revolution, and the Future of Programming*. (IEEE Annals of the History of Computing, Issue Vol. 36 No.04, pp: 64-73)
- Shasha, Dennis; Lazere, Cathy: *Out of Their Minds: The Lives and Discoveries of 15 Great Computer Scientists*. (Copernicus, 1998, 291pp)

References

1. Marateck, Samuel L. (1977). *FORTRAN* (Academic Press), p. 488
2. Courtois, Pierre-Jacques (2008) *Justifying the Dependability of Computer-based Systems: With Applications in Nuclear Engineering* (Springer), p. 112
3. Denning, Peter J.; Martell, Craig H. (2015) *Great Principles of Computing* (MIT Press), p. 157
4. Birrell, N. D.; Ould, M. A. (1985) *A Practical Handbook for Software Development* (Cambridge University Press), p.181
5. Haigh, Thomas (2010)
6. Albin, Stephen T (2003). *The Art of Software Architecture: Design Methods and Techniques* (Wiley Publishing, Inc.), p. 3
7. Hoare, C. A. R. (12 October 2010). "The 2010 Edsger W Dijkstra Memorial Lecture: What Can We Learn from Edsger W. Dijkstra?". Department of Computer Science, The University of Texas at Austin Retrieved 12 August 2015.
8. Ryder, Barbara G.; Sofa, Mary Lou; Burnett, Margaret (2005). *Impact of Software Engineering Research on Modern Programming Languages* ACM Transactions on Software Engineering and Methodology Vol. 14, No. 4, October 2005, p. 431-477.
9. Wirth, Niklaus (2008). *A Brief History of Software Engineering*. IEEE Annals of the History of Computing, vol.30, no. 3, July–September 2008, pp. 32–39.
10. In his 2004 memoir "A Programmer's Story: The Life of a Computer Pioneer", Brinch Hansen wrote that he used "Cooperating Sequential Processes" to guide his work implementing multiprogramming on the RC 4000, and described it saying, "One of the great works in computer programming, this masterpiece laid the conceptual foundation for concurrent programming."
11. As Lamport (2002) wrote, "Edsger W Dijkstra started the field of concurrent and distributed algorithms with his 1965 CACM paper "Solution of a Problem in Concurrent Programming Control", in which he first stated and solved the mutual exclusion problem. That paper is probably why POC exists; it certainly inspired most of my work."

12. Lamport, Leslie "Turing Lecture: The Computer Science of Concurrency: The Early Years (Communications of the ACM, Vol. 58 No. 6, June 2015)". ACM. Retrieved 22 September 2015.
13. Lo Russo, Graziano (1997). "An Interview with A. Stepanov (Edizioni Infomedia srl)". STLport. Retrieved 30 August 2015.
14. Hoare, Tony (March 2003). "Obituary: Edsger Wybe Dijkstra". *Physics Today*. **56** (3): 96–98. doi:10.1063/1.1570789
15. Faulkner, Larry R.; Durbin, John R. (19 August 2013). "In Memoriam: Edsger Wybe Dijkstra". The University of Texas at Austin. Retrieved 20 August 2015.
16. O'Regan, Gerard (2013). *Giants of Computing: A Compendium of Select, Pivotal Pioneers* (London: Springer Verlag), pp. 91–92
17. Apt, Krzysztof R. (2002)
18. Gries, David (1978). *Programming Methodology: A Collection of Articles by Members of IFIP WG2.3* (New York: Springer Verlag), p. 7.
19. Markoff, John (10 August 2002). "Edsger Dijkstra: Physicist Who Shaped Computer Era". NYTimes.com. Retrieved 10 April 2015.
20. Schofield, Jack (19 August 2002). "Edsger Dijkstra: Pioneering computer programmer who made his subject intellectually respectable". The Guardian. Retrieved 19 April 2015.
21. Knuth, Donald (1974). *Structured Programming with Go To Statements*. *Computing Surveys* 6 (4): 261–301. doi:10.1145/356635.356640.
22. Broy, Manfred; Denert, Ernst (eds.) (2002) *Software Pioneers: Contributions to Software Engineering*, p. 19. (Springer)
23. Nakagawa, Toru (18 July 2005). "Software Engineering And TRIZ (1) - Structured Programming Reviewed With TRIZ". TRIZ Journal. Retrieved 18 August 2015.
24. Hashagen, Ulf; Keil-Slawik, Reinhard; Norberg, A. (eds.) (2002). *History of Computing: Software Issues (International Conference on the History of Computing, ICHC 2000 April 5–7, 2000 Heinz Nixdorf MuseumsForum Paderborn, Germany)*. (Springer), p. 106.
25. Henderson, Harry (2009). *Encyclopedia of Computer Science and Technology*, revised edition. (Facts on File, Inc.), p. 150
26. "PODC Influential Paper Award: 2002", *ACM Symposium on Principles of Distributed Computing*, retrieved 2009-08-24
27. *Edsger W. Dijkstra Prize in Distributed Computing (Symposium on Principles of Distributed Computing – PODC)* retrieved 2015-08-01
28. *Edsger W. Dijkstra Prize in Distributed Computing (European Association for Theoretical Computer Science – EATCS)*, retrieved 2015-08-01
29. *Edsger W. Dijkstra Prize in Distributed Computing (International Symposium on Distributed Computing – DISC)* (<http://www.disc-conference.org/wp/dijkstra-prize/>)
30. "Edsger Wybe Dijkstra". *Stichting Digidome* 3 September 2003. Archived from the original on 6 December 2004.
31. O'Connor, J J; Robertson, E F (July 2008). "Dijkstra biography". *The MacTutor History of Mathematics, School of Mathematics and Statistics, University of St Andrews, Scotland*. Archived from the original on 11 October 2013. Retrieved 18 January 2014.
32. E. W. Dijkstra Archive (<http://www.cs.utexas.edu/users/EWD/transcriptions/EWD03xx/EWD340.html>)
33. E. W. Dijkstra Archive (<http://www.cs.utexas.edu/users/EWD/transcriptions/EWD03xx/EWD340.html>)
34. James, Mike (1 May 2013). "Edsger Dijkstra - The Poetry of Programming". i-programmer.info. Retrieved 12 August 2015.
35. Goodwins, Rupert (8 August 2002). "Computer science pioneer Dijkstra dies". Retrieved 22 December 2010.
36. Laplante, Phillip A. (ed.) (1996). *Great Papers in Computer Science* (Minneapolis-St. Paul: West Publishing Company)
37. Chen, Peter P. (2002). *From Goto-less to Structured Programming: The Legacy of Edsger W. Dijkstra*. (IEEE Software, 19[5]: 21)
38. Laplante, Phillip A. (2008). *Great Papers in Computer Science: A Retrospective*. (Journal of Scientific and Practical Computing, Vol. 2, No. 1, December 2008, pp 31–35)
39. 2002 PODC Influential Paper Award (ACM Symposium on Principles of Distributed Computing) (<http://www.podc.org/influential/2002-influential-paper/>)
40. Hart, P. E.; Nilsson, N. J.; Raphael, B. (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". *IEEE Transactions on Systems Science and Cybernetics* **SSC-4** (2): 100–107. doi:10.1109/TSSC.1968.300136
41. Frana, Philip L. (2001). "An Interview with Edsger W. Dijkstra (OH 330)". *Communications of the ACM* **53** (8): 41–47. doi:10.1145/1787234.1787249.
42. Dijkstra biography (<http://www-history.mcs.st-and.ac.uk/Biographies/Dijkstra.html>)
43. Jarník, V. (1930), "O jistém problému minimálním" [About a certain minimal problem], *Práce Moravské Přírodovědecké Společnosti* (in Czech), **6**: 57–63.
44. Prim, R. C. (November 1957), "Shortest connection networks And some generalizations", *Bell System Technical Journal*, **36** (6): 1389–1401, doi:10.1002/j.1538-7305.1957.tb01515.x
45. Dijkstra, E. W. (1959), "A note on two problems in connexion with graphs" (PDF), *Numerische Mathematik* **1**: 269–271, doi:10.1007/BF01386390

46. Pettie, Seth; Ramachandran, Vjaya (2002), "An optimal minimum spanning tree algorithm"*Journal of the ACM*, **49** (1): 16–34, doi:10.1145/505241.505243 MR 2148431.
47. MR 34/61 (<https://repository.cwi.nl/noauth/search/fullrecord.php?publnr=9251>)
48. Dolev, Shlomi (2000). *Self-stabilization* (MIT Press), p. 16
49. "People Behind Informatics: Dijkstra-Breakthroughs, Glossary (Garbage Collection)". University of Klagenfurt Retrieved 12 August 2015.
50. Hudson, Richard (31 August 2015). "Go GC: Prioritizing low latency and simplicity". The Go Programming Language Blog. Retrieved 21 September 2015.
51. van Emden, Maarten (6 May 2008). "I remember Edsger Dijkstra (1930–2002)". Retrieved 22 December 2010.
52. Hoare, C.A.R. (December 1973). "Hints on Programming Language Design"(PDF). p. 27.
53. Stroustrup, Bjarne(2014). *Programming: Principles and Practice Using C++*, 2nd Edition. (Addison-Wesley Professional), p. 827
54. Gram, Christian; Rasmussen, Per; Østegaard, Søren Duus (eds.) (2015). *History of Nordic Computing 4 4th IFIP WG 9.7 Conference, HiNC 4, Copenhagen, Denmark, August 13–15, 2014, Revised Selected Papers*(Springer), p. 358
55. Daylight, E. G. (2011). "Dijkstra's Rallying Cry for Generalization: the Advent of the Recursive Procedure, late 1950s early 1960s". *The Computer Journal* doi:10.1093/comjnl/bxr002
56. Report about the NATO Software Engineering Conference dealing with the software crisis(<http://homepages.cs.ncl.ac.uk/brian.randell/NATO/NATOReports/index.html>)
57. Report on a conference sponsored by the NATO SCIENCE COMMITTEE Garmisch, Germany 7th to 11th October 1968 (<http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF>)
58. Dijkstra, Edsger W *A Case against the GO TO Statement (EWD-215)*(PDF). E.W. Dijkstra Archive. Center for American History University of Texas at Austin (transcription (<http://www.cs.utexas.edu/users/EWD/transcriptions/EWD02xx/EWD215.html>))
59. Dijkstra, E. W (March 1968). "Letters to the editor: go to statement considered harmful"*Communications of the ACM* **11** (3): 147–148. doi:10.1145/362929.362947. ISSN 0001-0782
60. Knuth, Donald(1974)
61. Mills, Harlan D.(1986). *Structured Programming: Retrospect and Prospect*. (IEEE Software 3(6): 58-66, November 1986).
62. Reilly, Edwin D. (2004). *Concise Encyclopedia of Computer Science*(John Wiley & Sons, Ltd.), p. 734.
63. Meyer, Bertrand (2009). *Touch of Class: Learning to Program Well with Objects and Contracts* (Springer), p. 188.
64. Meyer, Bertrand (2009), p. 188
65. Reilly, Edwin D. (2004), p. 734.
66. Graba, Jan (1998). *Up and Running with C++*. (Springer), p. 1
67. Broy, Manfred; Denert, Ernst (2002)
68. Henderson, Harry (Facts on File, Inc., 2009)
69. Selby, Richard W. (2007). *Software Engineering: Barry W Boehm's Lifetime Contributions to Software Development, Management, and Research*. (IEEE Computer Society), pp. 701–702
70. Dijkstra, Edsger W (1982). "On the role of scientific thought." *Selected writings on Computing: A Personal Perspective* New York, NY, USA: Springer-Verlag. pp. 60–66. ISBN 0-387-90652-5
71. Brown, Kyle; Craig, Gary; Hester Greg (2003).
72. Grier, David Alan. "Closer Than You Might Think: Layers upon Layers". IEEE Computer Society Retrieved 12 August 2015.
73. Albin, Stephen T (Wiley Publishing, Inc., 2003), p. 3
74. Dijkstra, Edsger W *Over seinpalen (EWD-74)*(PDF). E.W. Dijkstra Archive. Center for American History University of Texas at Austin (transcription (<http://www.cs.utexas.edu/users/EWD/transcriptions/EWD00xx/EWD74.htm>))
75. Brinch Hansen, Per(2002). *The Origin of Concurrent Programming: From Semaphores to Remote Procedure Calls*. (Springer)., p. 8
76. McCormick, John W; Singhoff, Frank; Hugues, Jérôme (2011). *Building Parallel, Embedded, and Real-Time Applications with Ada*(Cambridge University Press), p. 5.
77. Anderson, J.H.; Kim, Y-J.; Herman, T (2003). *Shared-Memory Mutual Exclusion: Major Research Trends Since 1986* (Distributed Computing 16(2-3), 75-10)
78. Alagarsamy, K. (2003). *Some Myths About Famous Mutual Exclusion Algorithms*(ACM SIGACT News, 34(3): 94-103)
79. Raynal, Michel (2013). *Concurrent Programming: Algorithms, Principles, and Foundations*(Springer), p. vi.
80. James, Mike (1 May 2013). "Edsger Dijkstra - The Poetry of Programming". i-programmer.info. Retrieved 12 August 2015.
81. Hoare, C. A. R. (2004). "Communicating Sequential Processes"(PDF). Prentice Hall International.

82. Edsger W. Dijkstra Prize in Distributed Computing (ACM Symposium on Principles of Distributed Computing)(<http://www.podc.org/dijkstra>). The citation for the prize reads: "The Edsger W. Dijkstra Prize in Distributed Computing is named for Edsger Wybe Dijkstra (1930–2002), a pioneer in the area of distributed computing. His foundational work on concurrency primitives (such as the semaphore), concurrency problems (such as mutual exclusion and deadlock), reasoning about concurrent systems, and self-stabilization comprises one of the most important supports upon which the field of distributed computing is built. No other individual has had a larger influence on research in principles of distributed computing."
83. Edsger W. Dijkstra Prize in Distributed Computing (EATCS International Symposium on Distributed Computing)(<http://www.eatcs.org/index.php/dijkstra-prize>)
84. 2002 PODC Influential Paper Award (ACM Symposium on Principles of Distributed Computing) (<http://www.podc.org/influential/2002-influential-paper/>)
85. Dolev, Shlomi (2000). *Self-stabilization* (MIT Press), p. 3
86. Back, Ralph-Johan; Wight, Joakim (1998). *Refinement Calculus A Systematic Introduction (Texts in Computer Science)*. (Springer)
87. Morgan, Carroll; Vickers, Trevor (1992). *On the Refinement Calculus* (Springer)
88. Back, Ralph-Johan; Wight, Joakim (1998), p. v
89. Stabler, Edward P. (2014-01-01). Roeper, Tom; Speas, Margaret, eds. *Recursion in Grammar and Performance Studies in Theoretical Psycholinguistics*. Springer International Publishing. pp. 159–177. doi:10.1007/978-3-319-05086-7_8 ISBN 978-3-319-05085-0
90. Dijkstra, Edsger W *On a cultural gap* (EWD-924)(PDF). E.W. Dijkstra Archive. Center for American History University of Texas at Austin (transcription(<http://www.cs.utexas.edu/users/EWD/transcriptions/EWD09xx/EWD924.tml>))Dijkstra, E.W. (1986). "On a cultural gap". *The Mathematical Intelligencer* **8** (1): 48–52. doi:10.1007/bf03023921
91. Dijkstra, Edsger W *On the cruelty of really teaching computer science* (EWD-1036)(PDF). E.W. Dijkstra Archive. Center for American History University of Texas at Austin (transcription(<http://www.cs.utexas.edu/users/EWD/transcriptions/EWD10xx/EWD1036.html>))
92. Irfan Hyder, Syed (2013)
93. *In Memoriam Edsger Wybe Dijkstra* (memorial), University of Texas.
94. Istrail, Sorin (Brown University Department of Computer Science Alumni Magazine, v17.2, 2008)
95. Dijkstra, Edsger "Denken als Discipline". VPRO. Noorderlicht Retrieved 21 June 2016.
96. Dijkstra, Edsger W *How do we tell truths that might hurt?* (EWD-498)(PDF). E.W. Dijkstra Archive. Center for American History University of Texas at Austin (transcription(<http://www.cs.utexas.edu/users/EWD/transcriptions/EWD04xx/EWD498.html>))
97. Dijkstra, Edsger W *EWD-475* (PDF). E.W. Dijkstra Archive. Center for American History University of Texas at Austin. (transcription(<http://www.cs.utexas.edu/users/EWD/transcriptions/EWD04xx/EWD475.html>))
98. Dijkstra, Edsger W *EWD-539* (PDF). E.W. Dijkstra Archive. Center for American History University of Texas at Austin. (transcription(<http://www.cs.utexas.edu/users/EWD/transcriptions/EWD05xx/EWD539.html>))
99. Dijkstra, Edsger W *EWD-427* (PDF). E.W. Dijkstra Archive. Center for American History University of Texas at Austin. (transcription(<http://www.cs.utexas.edu/users/EWD/transcriptions/EWD04xx/EWD427.html>))
100. Dijkstra, Edsger W *EWD-443* (PDF). E.W. Dijkstra Archive. Center for American History University of Texas at Austin. (transcription(<http://www.cs.utexas.edu/users/EWD/transcriptions/EWD04xx/EWD443.html>))
101. Dijkstra, Edsger W (1982). *Selected Writings on Computing: A Personal Perspective* Berlin: Springer-Verlag. ISBN 978-0-387-90652-2
102. *Online EWD archive*, University of Texas.
103. Woehr, Jack (1 April 1996). "An interview with Donald Knuth". Dr. Dobb's Journal Retrieved 12 August 2015.
104. Edsger W. Dijkstra - Award Winner - ACM Awards (http://awards.acm.org/award_winners/dijkstra_1053701.cfm) (Extract from the Turing award Citation ready by M. Doug McIlroy chairman of the ACM Turing Award Committee, at the presentation of his lecture on August 14, 1972, at the ACM Annual Conference in Boston.)
105. Dale, Nell; Lewis, John (2011). *Computer Science Illuminated* 4th Edition (Jones and Barlett Publishers, LLC.), p. 313
106. Denning, Peter J. (2004). *The Field of Programmers Myth* (Communications of the ACM, 47 (7) pp. 15–20)
107. "Edsger Wybe Dijkstra (1930 - 2002)". Royal Netherlands Academy of Arts and Sciences Retrieved 17 July 2015.
108. "A. M. Turing Award". Association for Computing Machinery Retrieved 5 February 2011.
109. "Edsger W. Dijkstra 1974 Harry H. Goode Memorial Award Recipient". IEEE Computer Society Retrieved 17 January 2014.
110. "ACM Fellows – D". Association for Computing Machinery Retrieved 15 February 2011.
111. "Roll of Distinguished Fellows". British Computer Society Archived from the original on 4 March 2016 Retrieved 2014-09-10.
112. Feijen, W.; van Gasteren, A.J.M.; Gries, D.; Misra, J. (eds.) *Beauty Is Our Business: A Birthday Salute to Edsger W Dijkstra*. (Springer)
113. Awards: Loyola University Chicago(<http://www.luc.edu/cs/academics/awards/>)

External links


- E.W. Dijkstra Archive (<http://www.cs.utexas.edu/users/EWD/>) (Center for American History, University of Texas at Austin)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Edsger_W._Dijkstra&oldid=773498223"

Categories: 1930 births | 2002 deaths | Theoretical computer scientists | Computer science writers | Dutch computer scientists | Dutch computer programmers | Dutch mathematicians | Dutch physicists | Dutch science writers | Dutch academics | Eindhoven University of Technology faculty | Fellows of the Association for Computing Machinery | Fellows of the British Computer Society | Formal methods people | Computer programmers | Systems scientists | Leiden University alumni | Members of the Royal Netherlands Academy of Arts and Sciences | People from Rotterdam | Programming language designers | Programming language researchers | Researchers in distributed computing | Dutch software engineers | Software engineering researchers | Turing Award laureates | University of Texas at Austin faculty | Dijkstra Prize laureates | Deaths from cancer in the Netherlands | Burroughs Corporation people | Dutch expatriates in the United States | Deaths from colorectal cancer | Edsger W. Dijkstra



Wikimedia Commons has media related to ***Edsger Wybe Dijkstra***.



Wikiquote has quotations related to: ***Edsger W. Dijkstra***

- This page was last modified on 2017-04-03, at 03:54:09.
- Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.