Edsger Wybe Dijkstra
A life’s work making Programming a Professional Discipline

Nicola Holmes

Seminar report
UNIVERSITY OF HELSINKI
Department of Computer Science

Helsinki, April 25, 2016
Edsger Wybe Dijkstra contributed forty plus years of work in the field of Computer Science, shaping the discipline and developing many of the design principles and building blocks that make up what defines Computer Science. With an unwavering dedication to elegance in programming, Dijkstra was often focused on providing proofs of mathematical rigour to programming methods. Amongst his notable contributions to the field include the Shortest Path Algorithm now known as **Dijkstra's Algorithm**, the proponents of **Structured Programming**, the concept of the **Call Stack**, the programming concept of **Semaphore** and **Deadlock avoidance algorithms**.
Contents

1 Introduction 1

2 Biography 1
   2.1 The Student ........................................ 1
   2.2 The Programmer ...................................... 2
   2.3 The Research Fellow ................................. 4
   2.4 The Professor ........................................ 4
   2.5 The Wordsmith ....................................... 5
   2.6 The Retirement ....................................... 6

3 Select Significant Scientific Contributions 6
   3.1 Dijkstra’s Algorithm & DJP algorithm .............. 6
   3.2 Call Stack ............................................ 7
   3.3 Structured Programming ............................... 7
   3.4 Semaphore ............................................. 7
   3.5 Dining Philosophers Problem ........................ 7

4 Awards and Recognition 8

5 Conclusion 8

References 9
1 Introduction

When Edsger Wybe Dijkstra first decided to pursue a career in Computer Science, the idea of ‘programmer’ as a respectable profession was almost incomprehensible in his native Netherlands. Over his forty plus years of work in the field, Dijkstra helped shape Computer Science not only as a discipline, but developed many of the design principles and building blocks that make up what defines Computer Science, and provided proofs of scientific rigour to its academic field. Each of his many publications, treatises and commentaries are written with an economic candour that clearly show his wit, eloquence and deep understanding of the topics he discusses.

Dijkstra’s career was shaped by many factors, including a willingness to learn and to take advice from his scientific colleagues, and a passion for teaching. He insisted upon elegant simplicity in programming and mathematics, and was dedicated to developing and applying the rigour of mathematical arguments to programming. Amongst his notable contributions to the field include the Shortest Path Algorithm now known as Dijkstra’s Algorithm, the proponents of Structured Programming, the concept of the Call Stack, the programming concept of Semaphore and Deadlock avoidance. In an early paper involving the development of an operating system for an early digital computer, Dijkstra made a comment that would hold true for the correspondence for the rest of his career: "Be aware of the fact that experience does by no means automatically lead to wisdom and understanding"[13]

2 Biography

2.1 The Student

Edsger Wybe Dijkstra was born in Rotterdam on May 11th, 1930, the third of four children to Douwe Wybe Dijkstra and Brechtje Cornelia Kluijver. [19] His father, Douwe, taught chemistry at a Rotterdam secondary school and was later made its superintendent. Although he was well-known - with a Presidency of the Dutch Chemical Society - he was uninterested in scientific promotion. Dijkstra’s mother Brechtje was a mathematician, who like many women of her day did not have a job despite having a college education. Dijkstra describes her as as having "a great agility in manipulating formulae and a wonderful gift for finding very elegant solutions".[10]

Dijkstra initially considered pursuing a career in Law, with the goal to represent the Netherlands in the United Nations. Fortunately for the fields of Computer Science and Software Engineering, he was convinced otherwise. Despite being sure that he would fail his final exams at the Gymnasium in 1948, he instead achieved some of the highest marks that had ever been seen in the sciences by his teachers. Thus convinced, Dijkstra studied mathematics and theoretical physics at Leiden University.[10]
In 1951, Dijkstra’s father read an advertisement in *Nature Magazine* for a three-week course in Cambridge, teaching programming for the EDSAC electronic computer. He offered Dijkstra the course as a reward for passing his mid-year exam early. With the belief that knowing how to use an electronic computer "might come in handy", Dijkstra attended the course in September 1951.[10] The Director of the Computation Department at the Mathematisch Centrum (Mathematical Centre) in Amsterdam, Adriaan van Wijngaarden, had attended the same course one year earlier, and upon hearing of Dijkstra’s attendance invited him to visit the Centrum. In March 1952, Dijkstra accepted a job from van Wijngaarden to become the first official programmer in the Netherlands.[6]

After the death of his supervisor at Leiden, Dijkstra found himself caught between programming in Amsterdam at the Mathematisch Centrum, or seriously pursuing his theoretical physics in Utrecht.[10] It was a talk with van Wijngaarden that convinced him to finish his studies in theoretical physics with a minimum of effort in order to pursue a career in programming. Dijkstra said of this in his Turing Award Speech in 1972, "One moral of the above story is, of course, that we must be very careful when we give advice to younger people: sometimes they follow it!"[6] Upon graduation in 1956, Dijkstra moved to Amsterdam to work full-time at the Mathematisch Centrum.[10]

### 2.2 The Programmer

Although ‘programmer’ was not an accepted professional discipline at the time - Dijkstra’s 1957 marriage record shows ‘theoretical physicist’ despite his objections - van Wijngaarden had put the idea into Dijkstra’s mind that he could be one of the people to make it so. His experiences at the Mathematisch Centrum in Amsterdam put him on the path to recognising what this relatively new field would require of him in order for this to happen. Dijkstra collaborated with Bram J. Loopstra and Carel S. Scholten, who had been employed to build a computer for the Centrum, the ARMAC. In this collaboration, Dijkstra learnt the importance of clear documentation and careful design in avoiding programming bugs and technological mistakes.[18]

It was during this time that Dijkstra developed his *Shortest Path Algorithm*, although he would not publish it until 1959. This pivotal non-trivial algorithm of computer science was developed as a simple demonstration of the ARMAC’s power for its official inauguration in 1956. This algorithm is now known as *Dijkstra’s Algorithm* and it was created without pen nor paper in 20 minutes as he sat one sunny afternoon enjoying a coffee in the sun with his wife on a cafe terrace in Amsterdam.[10]

Dijkstra received his PhD from the University of Amsterdam in late 1959, for his thesis *Communication with an Automatic Computer*. This thesis described the assembly language of the Electrologica X1, which Loopstra and
Scholten had designed with a real-time interrupt. Dijkstra made the core problem of his thesis the development of the assembly language including a real-time interrupt handler for this computer.\[10\] It was during the development of the X1 that Dijkstra also developed his *Minimum Spanning Tree* algorithm. It was developed in order to minimise the copper wire required to connect points on the back of the X1’s wiring panel.\[20\] Both the *Minimum Spanning Tree* and *Shortest Path* algorithms were published in the first issue of *Numerische Mathematik* in 1959 as ‘A Note on Two Problems in Connexion with Graphs’.\[1\]

While still at the Mathematisch Centrum in 1959, Dijkstra became involved with the development of the high-level programming language ALGOL-60. He was tasked, along with JA Zonneveld, to develop a compiler for this system. Though initially instructed by van Wijngaarden to base this compiler on the work of experimental programmer Henry D Huskey - Dijkstra describes this as 'a chaotic mess' - Zonneveld and Dijkstra were convinced they had a better idea.\[16\] Writing in duplicate, and making multiple copies, the first compiler of ALGOL-60 was developed using new methodologies which are described in the May 1960 second issue of *Numerische Mathematik*, as 'Recursive Programming'.\[10\] This paper describes their implementation, while also introducing Dijkstra’s concept of a programming *Call Stack*.\[2\]

In 1962, Dijkstra took up his first teaching position as a Professor of Mathematics at the Eindhoven University of Technology. Although not the Department’s first choice - "because of my sandals, my beard and my arrogance" - Dijkstra taught Numerical Analysis while growing a group of computer scientists to work with on his next project.\[10\] This project was the *THE Multiprogramming System* for the Electrologica X8, described in the 1968 paper 'The structure of the “THE” multiprogramming system'.\[13\] This early Operating System was designed part-time along with five other early computer scientists, and set in place design principles which are now considered industry standards for professional programming. *THE* is recognised as the earliest example of software architecture. It includes levels of abstraction, programming in layers, semaphore and cooperating sequential processes.\[13\] Sadly, the Department did not see the significance of this system at the time, and Dijkstra’s research group was disbanded.\[10\]

Faced with an uncertain future, and experiencing depression, Dijkstra collected his thoughts from his experience of programming and sent copies to about twenty friends for comments.\[10\] These collected thoughts explored programming as a mathematical discipline, advocating design principles to apply rigour and mathematical logic to software design. What had been a way to clear his mind became the seminal publication 'Notes on Structured Programming' which was published in 1970.\[15\] Dijkstra was surprised by the quick spread of his notes, but acknowledges that it was 'the first text that openly took the programming challenge very seriously.'\[10\] In 1972, Dijkstra was awarded the ACM A.M. Turing Award. His acceptance speech, ‘The
Humble Programmer’, is also considered important reading for students of the discipline.[18] The lack of enthusiasm showed by his Department’s Chair at this award, coupled with their determination to ignore programming for 'genuine mathematics', Dijkstra left the University.[10]

2.3 The Research Fellow

In 1973, Dijkstra took up a position as a Research Fellow at the Burroughs Corporation. Working from his home in Neunen, Netherlands, Dijkstra was able to focus on pursuing his ideas and thoughts on the development of programming as a scientific field. Famous in the field of Computer Science by this time, Dijkstra was able to guest lecture in Universities throughout the world. He used these trips, and his visits to the Burroughs Research Facility in the USA, to find younger scientists to mentor, to engage with his scientific colleagues and also to improve his English skills.[18]

Pursuing the formal derivation of programs, Dijkstra developed the first self-stabilizing systems and also designed predicate transformers to define program semantics. These publications led to the seminal 1976 book, 'A Discipline in Programming'.[10] This focus on formal verification was concerned with streamlining the mathematical arguments behind the proofs of programming algorithms, allowing correctness by construction.[20] This book is considered a Citations Classic by the Science Citation Index.

Another significant contribution Dijkstra made during his time at Burroughs, was his development of the theory of non-determinacy. Non-determinism is when on multiple runs of the same program, different results are achieved. Dijkstra’s implementation of guarded commands and theory of non-determinacy allows otherwise trivially different programs to be mapped together, allowing programs to derived in a more systematic manner.[7] This is the basis of the design principle that allows system components in code to be replaced by components that run at different speeds, while not affecting the overall correctness of the system design.[18]

It was also during Dijkstra’s time at Burroughs that almost 500 of his EWD series were written.[20] The EWD manuscripts are the correspondence of Dijkstra with his peers, and are a collection of notes, observations and commentaries that span almost forty years. Many of Dijkstra’s publications began as EWD manuscripts, and these short documents - mostly handwritten - give a unique insight into his personality and thought processes.[18]

2.4 The Professor

When in the early 1980s the Burroughs Corporation changed managements - and thereby focus - Dijkstra accepted the position of the Schlumberger Centennial Chair in Computer Sciences at the University of Texas in Austin. Although taking his role as a teacher and mentor very seriously, Dijkstra
continued to be prolific in his research regarding formal verification and communications on the nature of computer science as a discipline. In this time he published two notable books, the first a primer for students of Computer Science 'Een methode van programmeren/A Method of Programming' and the second with Carel S. Scholten, 'Predicate Calculus and Program Semantics'. The latter is devoted to analysis of Dijkstra’s *weakest precondition semantics*.

Beginning from his time at Eindhoven, Dijkstra had put great thought into how best teach Computer Science. Never following textbooks, Dijkstra would write the notes for the lectures directly on the blackboard and encourage his students to suggest ideas. Final exams were conducted orally over several hours, in Dijkstra’s office or home. As he would take a photo of each student at the start of the semester to remember their names, he would then return these to each student signed after their exam with a beer and a talk about their learning experience.[18]

Dijkstra had strong opinions that computer programming should be considered a branch of mathematics, and even objected to the very term 'Computer Science', preferring 'Computing Science' as a descriptive term.[14] He believed that the abstract mechanisms of complexity should be the core of Computer Science, not the technology. He was concerned that academia and industry believed the theoretical side of Computer Science was over, and all that was left were engineering practicalities.[12] He also considered, for the record, that Software Engineering should be known as "The Doomed Discipline", with the charter "How to program if you cannot".[14]

2.5 The Wordsmith

In his career, Dijkstra wrote nine books, twelve book chapters, forty journal articles, thirty-four contributions to conference proceedings, twenty-two miscellaneous publications and 1318 EWDs.[18] There is little argument from anyone who has read a word from Dijkstra’s hand that he had a irrepressible personality. His unwavering confidence in his ideas comes through in both his Dutch and English texts, and many stories of his unconventional approaches abound. He was known for writing in an economic fashion that remained eloquent while being simplistic. Many of his EWD manuscripts show his humour directly, and even his larger texts display a way with words that make understanding complex theories of computer science more simple. A good example of this is the quote 'The question of whether Machines Can Think... is about as relevant as the question of whether Submarines Can Swim.' from his 1984 EWD 'The Threats to Computing Science'.[8]

Though supportive of many of his peers, Dijkstra never seemed to pause before delivering clear wit at the failings of others. His direct honest wit-ticisms are well known in the field, with a clear example of this being his 1968 open letter sent to the *Communications of the ACM*, 'GO TO Statement Considered Harmful' (originally titled: 'A Case against the GO TO
Statement’). This letter opens up with: '...the quality of programmers is a decreasing function of the density of go to statements in the programs they produce'.[4] This letter sparked debate in the programming community - including a response from Frank Rubin entitled 'GO TO Considered Harmful Considered Harmful' - which led to Dijkstra’s reply 'On a Somewhat Disappointing Correspondence'. This reply elegantly shreds Rubin’s response, then concludes with 'The whole correspondence was carried out at a level that vividly reminded me of the intellectual climate of twenty years ago, as if stagnation were the major characteristic of the computing profession, and that was a disappointment.'[9]

2.6 The Retirement

In November 1999, Dijkstra retired from active teaching and on his 70th birthday in May 2000, the Department of Computer Sciences at the University of Texas held a symposium to celebrate his seminal contributions to Computer Science. Retiring home to Nuenen in the Netherlands, he died in August 2002.

3 Select Significant Scientific Contributions

3.1 Dijkstra’s Algorithm & DJP algorithm

Dijkstra took simplicity seriously, so one should note that these two algorithms were first published in less than two and a half pages.[1] The background surrounding this publication is explored in Section 2.2. Dijkstra’s Algorithm was described by him in correspondences as the Shortest Path Algorithm.[10]

Dijkstra’s Algorithm finds the shortest path between two nodes, in a set of \( n \) nodes connected by paths, by first selecting the unvisited node with the shortest path. The distance to each unvisited neighbour node to this initial node is then calculated. If this distance is smaller, the distance of the neighbour is updated to consider the path already travelled. Once all the neighbours of the initial node have been calculated, the node is set to visited and the process is repeated. Dijkstra’s method requires the storing of the information for less than \( n \) paths simultaneously.

The DJP or Prim’s Algorithm finds a Minimum Spanning Tree by building the tree one vertex at a time, adding the cheapest possible connection from the tree to another vertex, in essence finding the shortest path at each vertex to another unknown vertex.
3.2 Call Stack

Dijkstra introduced the concept of the Stack (or Call Stack) in the 1960 publication ‘Recursive Programming’. A Stack is a system for storing a sequence of information units that is increased or decreased at one end only. Successive storage locations are set aside, and a stack pointer is defined which always points to the first free ‘place’.\[2\] Now considered a fundamental part of programming and computer science, this device was first developed by Dijkstra to assist in programming his ALGOL-60 compiler.

3.3 Structured Programming

It was Dijkstra who coined the term Structured Programming in his 1968 open letter described in Section 2.5, and it is often known now as GoTo-less Programming. Though he did not invent the concept, Dijkstra was a major contributor to the development of the movement with his ‘Notes on Structured Programming’ and subsequent book with C.A.R. Hoare and O-J Dahl, ‘Structured Programming’. As a paradigm, Structured Programming avoids simple tests and jumps by implementing subroutines, for and while loops, and block structures. Examples of codes influenced by this paradigm include Pascal, C, Modula-2, Ada, PL/I and C++, and the paradigm Procedural Programming is a derivation.\[17\]

3.4 Semaphore

Dijkstra’s Semaphore mechanism was developed to provide mutual exclusion for \(n\) processes, and is considered one of the first published concurrent algorithms.\[3\] In practice, a semaphore is a variable that keeps track of the availability a resource at the programmers discretion. This can be a counting or a binary variable. A trivial counting semaphore is implemented with incremental \(V\) and decremental \(P\) operations.\[3\]

3.5 Dining Philosophers Problem

Originally developed as a student exam problem while teaching at Eindhoven, this problem was initially called the Dining Quintuple Problem.\[5\] The problem is that there are five philosophers sitting around a table each with a bowl of difficult spaghetti that requires two forks, but only one fork each. A fair method must be found that allow all the philosophers to eat, and prevents a deadlock situation. Dijkstra’s original solution is known as the Resource Hierarchy Solution. It gives each philosopher a fair turn with two forks to eat when they are hungry. This uses a private semaphore system for each process which creates a partial order convention that all resources are requested in order, therefore preventing a starving philosopher and a deadlock.\[5\]
4 Awards and Recognition

In 1972, Dijkstra was called from the USA by Franz L. Alt of the ACM to inform him that he had won the ACM A.M. Turing Award. Despite being known for his arrogance, Dijkstra was humbled and overwhelmed.[10] He was the first award winner to not be a native English speaker. The award citation reads: "For fundamental contributions to programming as a high, intellectual challenge; for eloquent insistence and practical demonstration that programs should be composed correctly, not just debugged into correctness; for illuminating perception of problems at the foundations of program design."[20]

A two-day seminar on Computer Science was organised in 1990 at the University of Texas in Austin to mark Dijkstra's 60th birthday. Since 2010, the Department of Computer Science there holds an Edsger W. Dijkstra Memorial Lecture.[18]

Presented annually since 2000, and won by Dijkstra in 2002 before his death, the PODC Influential-Paper Award was renamed the Edsger W. Dijkstra Prize in Distributed Computing in March 2003. It is awarded to outstanding papers which have had a significant influence on distributed computing for at least a decade.

Dijkstra also has a significant list of awards, memberships, fellowships, recognitions and honorary doctorates to his name.[19]

5 Conclusion

A husband, a father, and a lover of Mozart, it is hard to see how Computer Science as we know it today would be without the shaping force of Edsger Wybe Dijkstra for over forty years. With the impact he has had on a discipline that he almost didn’t enter twice, we can only imagine where the fields of Dutch International Relations or of Theoretical Physics would be if he had not changed course. With an unwavering dedication to the simple elegance of programming algorithms, he was never interested in writing in modern coding languages but instead in making the theoretical computing science behind them better. Dijkstra has had a lasting impact on the development and the teaching methods of the discipline of Computer Science.

In 1995, he summed up his contribution with candour for someone who was never overly concerned with principles of modesty or mincing words: "I mean, if 10 years from now, when you are doing something quick and dirty, you suddenly visualize that I am looking over your shoulders and say to yourself 'Dijkstra would not have liked this.', well, that would be enough immortality for me."[11]
References


