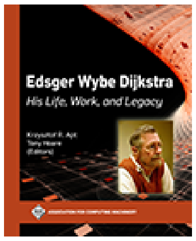
Review of [1]

Edsger Wybe Dijkstra – His Life, Work, and Legacy
by **Krzysztof R. Apt and Tony Hoare** (Editors)

ACM Books and Morgan & Claypool Publishers, 2022
Paperback, 550 pages, $ 72.00



Review by **Sarvagya Upadhyay**
(`supadhyay@fujitsu.com`)
Fujitsu Research of America
4655 Great America Parkway, Suite 410
Santa Clara CA 95054, USA

1 Overview

Edsger Wybe Dijkstra (1930–2002) was a Dutch computer scientist known for his significant contributions to theoretical computer science, programming, and software engineering. Always interested in mathematics and physics, Dijkstra briefly explored theoretical physics before ultimately completing his doctoral studies in computer science.

Dijkstra is widely recognized for his eponymous single-source shortest-path algorithm, formal advancements in computer programming, and foundational work in program verification and structured programming. One of his most famous stances is his vehement opposition to GOTO statements, a view that is widely accepted by generations of programmers that followed him and demonstrates his commitment to clear and structured code.

Dijkstra's contributions to computer science earned him numerous accolades, most notably the ACM Turing Award in 1972. His 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." His influence is so profound that there's an award named after him, the Edsger W. Dijkstra Prize in Distributed Computing.

This book showcasing his impact involves 31 computer scientists from diverse fields touched by his research, and is divided into five parts. The first part features Dijkstra's Turing Award speech, "The Humble Programmer", offering a glimpse into his philosophical and methodological approaches to programming and problem-solving. Following this, the next two parts delve into his specific contributions, with the second part exploring various facets of his research from the point of view of contemporary researchers, while the third part includes nine of his seminal publications. The fourth part delves into Dijkstra's life through biographical essays, highlighting his influence on the field and personal anecdotes that shed light on his character. Finally, the fifth part discusses Dijkstra's role as a lecturer, the E. W. Dijkstra archive, and provides concise professional biographical information.

[1] ©2024 Sarvagya Upadhyay

2 Summary

As mentioned in the previous section, this book consists of five parts and includes contributions from 31 computer scientists. The Preface of this book states that five contributors are themselves recipient of the prestigious ACM Turing award.

Part I: The Humble Programmer The book opens with Dijsktra's Turing award speech. I have revisited these 18 pages on a few occasions. It reflects his knack for communicating his thoughts in a clear and accessible manner. I really loved going through these 18 pages. Dijkstra received the Turing Award during the nascent stages of computer science, amidst a digital revolution. And although he may have not envisioned the evolution of different computing paradigms, as a quantum computer scientist, I find his speech highly relevant to the ongoing quantum revolution. This starts with the first page itself, where he discusses how his hardware counterparts could showcase a physical device and explain its workings to showcase their professional competence. I see quantum computing in a similar early stage. For instance, displaying a superconducting quantum computer, resembling an exquisite chandelier, can awe visitors. Yet, a researcher or developer working on quantum applications may not impress to the same extent.

Beyond biased perspectives from a quantum computer scientist, Dijkstra's speech contains valuable insights still applicable in today's computing landscape. For instance, his critique of the design of "third-generation computers" in the mid-sixties prompts us to consider optimizing hardware devices for maximum benefits. The concept of hardware accelerators or specialized processing units like GPUs aligns with Dijkstra's vision of computing design benefiting us through both raw computing power and ease of application development.

Finally, I find Dijkstra's vision of the future of programming particularly fascinating and somewhat prophetic. Each generational shift in programming and what can be achieved through it has historically been disruptive, catching the world off guard and eliciting a sense of awe. Dijkstra accurately predicted the necessity of such dramatic revolutions in programming, and the world has since recognized and adapted to these seismic shifts. Economic factors have also driven the need for greater programming efficiency. Additionally, the profession seems to have evolved towards what Dijkstra envisioned as "intellectually manageable" programming.

There are several thoughts that crosses through your mind while reading through his speech, and I will leave it to the readers to draw their own conclusion. It is rich in anecdotal history and wisdom that is still relevant to this era.

Part II: Technical Perspectives on Dijkstra's Work This part comprises of nine chapters each discussing a different aspect of Dijkstra's work.

1. This first chapter focuses on Dijkstra's single-source shortest-path algorithm, described by Mikkel Thorup in his own words as "one of the most famous and broadly used greedy algorithms." Thorup delves into the algorithm itself, the subsequent work it inspired over the past sixty years, and the different implementations. Despite the algorithm's simplicity and clarity, along with its relevance demonstrated through comparative analyses with newer works, I find the chapter quite dense. It contains a substantial amount of information into just five pages. I would have preferred a more extensive treatment of the topic.

2. The second chapter is written by Butler Lampson on programming concurrent systems. It delves into Dijkstra's influential ideas from the 1960s aimed at mitigating the inherent nondeterminism of concurrency. As mentioned in the chapter, Dijkstra thought about these topics when machines had only one CPUs and concurrency was almost non-existent. The chapter briefly explores concurrency in CPUs and distributed systems and how the principles such as atomicity and commuting can alleviate nondeterminism in concurrent operations.. It finally gives the classification on ways to program concurrency.

3. The third chapter, authored by Leslie Lamport, focuses on concurrent algorithms. Dijkstra published five papers on concurrent algorithms, which this chapter explores in some detail. The problems that Dijkstra developed concurrent algorithms for are listed in chronological order: (i) mutual exclusion problem that asks for synchronization among processes to perform critical sections of a program, (ii) self-stabilization devoted to fault-tolerance in concurrent algorithms, (iii) on-the-fly garbage collection that will perform garbage collection concurrently with the main program as opposed to the traditional method of doing it by interrupting the program, and (iv) termination detection in distributed computing. Lamport emphasizes Dijkstra's pivotal role as the driving force behind these papers, despite some of them being of collaborative nature with co-authors, and refers to each one of them as Dijkstra's algorithm.

4. The fourth chapter in this part of the book discusses on Dijkstra's work on self-stabilization. The fundamental problem that Dijkstra thought on was how can a distributed system attain correct behavior after being in an incorrect state. This chapter is authored by Ted Herman where he presents a description of Dijkstra's self-stabilizing algorithm, the impact of the work, and finally his thoughts on its significance. I enjoyed reading this chapter especially the conclusion and Herman's lighthearted reasoning of why "Dijkstra's project on self-stabilization was superbly successful."

4. The fifth chapter, written by Reiner Hähnle, touches upon the topic of program verification and Dijkstra's central role in this topic. Dijkstra's usage of the term *program verification* referred to the correctness of a program with respect to all possible executions. It starts with Dijkstra's thoughts on program verification and critique by researchers in the field. Dijkstra favored *correctness-by-construction* approach as opposed to *post hoc verification*, where the program is verified after construction. The chapter focuses on his most important contribution, the weakest precondition calculus. There are some interesting points that Hähnle raises in view of Dijkstra's legacy in this field, which is the final section of this chapter.

5. The sixth chapter by David Gries delves into Dijkstra's work into program correctness from both an anecdotal and technical perspective. Dijkstra's thought along the lines of how to write a program and its proof of correctness simultaneously as part of his program development process. Gries outlines four key themes that motivated Dijkstra's pursuit of correct programs based on their correspondence. Subsequent sections explore Dijkstra's work on structured programming interspersed with quotes from several of his scholarly work, his participation in 1968 NATO Conference on Software Engineering and reflections on the event, and his personal state of mind during a period when he "wrote over 25 EWDs (technical reports) that touched upon program development." The author then highlights a few of Dijkstra's work relevant to program correctness.

6. The seventh chapter, co-authored by Krzysztof R. Apt (one of the editors of the book) and Ernst-Rüdiger Olderog reviews "Dijkstra's contribution to nondeterminism by discussing the

relevance and impact of his guarded commands language proposal." The authors start with explaining nondeterminism in computer science and continue to discuss about *angelic nondeterminism*, where one identifies successes and failures, and only the former counts. This type of nondeterminism is heavily studied and continues to have great impact in computational complexity theory research. The authors subsequently explore guarded commands, and illustrate computing maximum of two numbers and Euclid's greatest common divisor algorithm in favor of guarded commands. They go on to describe how concurrent programs and nondeterministic programs can be linked via guarded commands and several other concepts related to them. They also explore subsequent developments influenced by Dijkstra's work.

7. In this chapter, Christian Lengauer explores Dijkstra's views on software construction through personal reflections and correspondences with the man himself. The chapter opens with anecdotal accounts that offer insights into Dijkstra's personality, including a memorable image of them on a tandem bike in Nuenen. Lengauer delves into Dijkstra's strong disagreements with Backus, his preferences such as favoring textual representations over pictorial ones, and his opinionated stance, which could be both advantageous and detrimental to his peers. The chapter concludes on a poignant note with Dijkstra's final words to Lengauer.

8. This chapter, co-authored by Robert van de Geijn and Maggie Myers, begins with the a few interactions between the former author and Dijkstra. They discuss about the goal-oriented programming advocated by Dijkstra and how it can be utilized to discover new algorithms in dense linear algebra. As an example, the authors first give a simple in-place algorithm for inverting an upper triangular matrix and how by "systematically identifying multiple loop invariants", a highly parallelizable alternative to the simple algorithm can be derived. The authors then summarize the impact of Dijkstra's and his peers on architecture and software development of numerical algorithms.

9. In the ninth chapter, focusing on Dijkstra's technical contributions, delves into calculational proofs. Penned by Vladimir Lifschitz, the chapter commences with a letter from Dijkstra illustrating how natural number properties can be demonstrated using the calculational style. This approach to proofs was detailed in Dijkstra's book "Predicate Calculus and Program Semantics," which he co-authored with Carel Scholten. Although I have encountered several concepts discussed in the book thus far, calculational proofs were entirely new to me. However, after going through the chapter, I can say that I now know more about calculational proofs.

10. Jayadev Mishra discusses Dijkstra's broad range of interests in mathematics and how he expressed his ideas and proofs with exceptional elegance in a series of notes known as EWDs. In an earlier chapter, Lengauer mentioned Dijkstra's fondness for fountain pens, and Mishra's chapter showcases some of Dijkstra's beautifully handwritten notes done with a fountain pen. Mishra focuses on a few theorems where Dijkstra came up with a very elegant argument to prove them. I was pleasantly surprised by his graph-theoretic arguments to prove Fermat's and Wilson's theorems in number theory, and the AM–GM inequality. This chapter was a delight to read.

Part III: Selected Papers This part contains nine of the most influential papers authored and co-authored by Dijkstra. All of the papers have been discussed in the previous part.

Part IV: Biographical Essays This section comprises several chapters authored by computer scientists who had personal and professional relationships with Edsger Dijkstra. Notable contributors include Krzysztof R. Apt, E. Allen Emerson, David Gries, Tony Hoare (the co-editor of the book), Brian Randell, and Fred B. Schneider. A dedicated chapter explores how Dijkstra was perceived by his friends, colleagues, and students. Contributors to this chapter include Lex Bijlsma, Ken Calvert, K. Mani Chandy, Eric C. R. Hehner, Wim H. Hesselink, Rajeev Joshi, Don Knuth, Alain J. Martin, Jayadev Mishra, David A. Naumann, J. R. Rao, Hamilton Richards, Mark Scheevel, and David Turner. The personal reminiscence and reflections from these researchers highlight the profound impact Dijkstra had on generations of computer scientists. I highly recommend reading these accounts as they portray a figure universally admired and respected.

Part V: Varia This is final part of the book consisting of two chapters. The first chapter in this part is devoted to Dijkstra as a key lecturer and director of the Marktoberdorf summer schools. The first summer school was held in 1970, and Dijkstra continued to be the director until 2000. Manfred Broy discussed in detail about these summer schools, and after reading it, I wish we had more such events that reach out to a wider audience. The final chapter (Hamilton Richards) is about the Edsger W. Dijkstra archive, an online repository of Dijkstra's unpublished manuscripts.

3 Opinion

Having gone through the book, it is evidently clear that preparing this book was a monumental task. I enjoyed reading the book. My favorite parts of the book was Dijkstra's Turing award speech and the biographical essays contributed by various researchers in the field. This differs significantly from a standard biography, where the perspective is usually limited to one or maybe two authors. The multiple biographical essays of Dijkstra offer diverse portraits, enriching the experience for readers curious about his life.

Dijkstra started working as a programmer way before programming was even considered a profession. He even alluded to it in his speech when discussing his wedding with his wife Ria and how his profession was changed to "theoretical physicist" by the municipal corporation of the town of Amsterdam. His impact on the field is profound, as emphasized in the technical chapters dedicated to his work. However, these chapters aren't just overflowing with praise; they also critically analyze his views, scholarly contributions, and thought processes. This critical perspective is expected from the authors of these chapters. They demonstrate the research trajectories Dijkstra initiated – some widely adopted and acknowledged, while others not receiving as much attention. It goes without saying that Dijkstra's writings were the result of meticulous thought. These chapters also underscore the inherent humility in research. They highlight both the successful and not so successful research endeavors of Dijkstra.

The technical chapters can be daunting for a lot of readers. The authors have made significant effort to write the chapters in a way that could reach to a wider audience in computer science. Readers should not read this book in anticipation of understanding everything. Rather they should read it to celebrate Edsger W. Dijkstra's life and work, and perhaps gain insights into how the brilliant mind of his worked. I would like to end this review by mentioning few quotes of Dijkstra that are pertinent even today. The below quote can be found in Chapter 10 of the book.

> Simplicity is a great virtue, but it requires hard work to achieve it and education to appreciate it. And to make matters worse: complexity sells better.

Or his views on the evolution of computer science as a field during his lifetime.

> Computer science is no more about computers than astronomy is about telescopes.
>
> The question of whether a computer can think is no more interesting than the question of whether a submarine can swim.