Conduit

A RESEARCH AND ALUMNI NEWS MAGAZINE
DEPARTMENT OF COMPUTER SCIENCE
BROWN UNIVERSITY

A Visit to Al Quds University

ALSO INSIDE:

- The Kanellakis Legacy Lives On
- Storytelling About Lighthouses

Storytelling About Lighthouses:

When Professor Dijkstra Slapped Me in the Quest for Beautiful Code

By **Sorin Istrail**

*** Arrogance in computer science * Empirical science? * Philosophy * Goddess Reason * A Critic's Dilemma * Pajamas algorithms * Pastiche pie prize**

THE SLAP

I finally met E.W.Dijkstra, the brilliant computer scientist whom I had admired for years, in Newport, Rhode Island. It was 1986, three years after my immigration to the United States from Romania, and Dijkstra, by then the Schlumberger Centennial Chair in Computer Sciences at the University of Texas at Austin, had invited me to join him for a day at the Summer School on Program Construction being held at Salve Regina University.

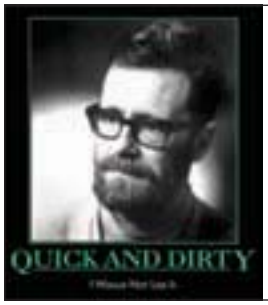
The day alternated between Professor Dijkstra lecturing and writing on a blackboard with his exquisitely precise handwriting. (Even now, I can recall how he returned to the blackboard to redraw a capital letter M that he felt lacked parallel vertical lines.)

At lunch, he sat near me at a long table with about ten others. Everybody there was eager to hear what this master would say, and because he spoke softly, few people if any were talking. We had just about finished eating when Dijkstra raised his voice to address me.

"Sorin - I have a problem for you."

"Sure," I said, thinking that this was the moment I had been waiting for since 1983, when I first solicited Dijkstra's comment and guidance on a technical report I had sent him previously.

The silence at the table became more pronounced.



"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." [1]

Edsger Dijkstra

"Suppose we play a two-player game played with identical coins on a table," Dijkstra began. "We have a bag of coins with as many as we need. The two players alternatively take a coin from the bag and place it on the table. The rules of the game forbid the coin to sit on top of another coin on the table, but it could hang off the table as long as it does not fall off. The player who puts the last coin on the table wins the game."

There was something in Dijkstra's expression – a combination of smile and an intense Don Quixotesque gaze – that seemed to raise the stakes at the table. I awaited *the* question.

"Is there an algorithm for one of the players to win always?" he asked with the quiet, calm assurance of a man who saw it all, who knew what would happen next.

I said nothing for a minute or two, anticipating the excitement that was to come. Then, as if jarred awake from a pleasant dream, I did what one does in such a moment: With my right hand, I reached down the neck of my sweater to remove a pen from my shirt pocket.

My hand was not even halfway there when – slap! Dijkstra smacked my hand away from its goal. Those few seconds that followed were long but I was slow to figure out, and I could not believe the strength of the slap. Instinctively, I drew away from him and the others at the table recoiled.

Dijkstra broke the silence. "Don't mess up your thoughts," he proclaimed. "Keep it simple, in your head."

It goes without saying that for the next few minutes I was useless at solving the puzzle and I bubbled nonsense. He waited a bit, then began offering clues. "Did I tell you anything about the table?"

"No," I responded. I began to think out loud. "As it must be true for all tables ..."

He interrupted with a question. "What is the smallest table?"

"One point?" I said trying to recover.

"Who wins then?" he asked.

By now rational, I responded. "The first player. So the first player always wins."

"Which table to consider next?"

"As big as a coin; again first player wins." I thought I was on a roll.

"No."

Me: "How about as big as two coins?"

Dijkstra: "Yes. Who wins then?"

Me: "First player puts his coin in the middle of the table. And I am not sure if the second can put his coin, but if the second can, so can the first again symmetrically; if not, neither can ... so first wins again!"

Dijkstra: "Do you have the algorithm now?"

Me (excitedly, knowing what he wanted to hear): "The invariant is Every point on the table, except the center, has a symmetric point with respect to the center."

Dijkstra: "Bravo."

(To translate the answer into *Dijkstranese*: The first player always puts his first coin on the center of the table. The second player puts his coin somewhere on the table that is free, and then the first player makes his move in exactly the symmetric point to the center, which is always free by the *invariant*. And so on, *mutatis mutandis*. The invariant assures the correctness of the algorithm no matter how many moves/how big the table.)

One of the Joys of Life and the Cruelty of Really Teaching Computing Science

Dijkstra was our Professor-in-Chief. How to teach computing science was one of the most fascinating life-long themes of reflection for Dijkstra. He wrote many articles about teaching methodology, he was the patriarch of posing beautiful problems and silly games, which gave him the opportunity to teach the art of problem solving. His algorithms, solutions of his games, define beautiful code, raising the derivation of the code from the program specification to an art performance. Those are not just beautiful games; they go to the heart of the difficulty of programming methodology, and illuminate hard-to-grasp complexities. It is an art form to invent such silly games; this art form should be encouraged and rewarded. They make the perfect opening of a lecture on the subject, especially because they satisfy the right axioms: unique solution, simplicity of algorithmic solution and rational step-by-step derivation of algorithm, generalizations become extremely difficult problems (try the above silly game with coins on a convex table), and most of all, your can find their solution without pen and pencil, in your head. I had the pleasure of attending some of Professor Dijkstra's lectures, to be trained one-on-one in the art, to have read many of his articles on teaching computing science, and to be inspired to uncover and present silly games occuring in everyday life, although their lessons are not as beautiful and important as those concocted by the Master of Silly Games. For the younger generation, here is a list of some his gems: the dining philosophers, the elephant made of mosquitoes humming in harmony, the toilet and trains, the plateau problem, the self-stabilization problem, average page fault frequency problem, the Dutch national flag, banker's algorithm, [the cryptographic game from[EWD 666]"a problem solved in my head."

Dijkstra examined radical innovation in teaching, and his thesis is always that universities should have guts in teaching.

> *"Teaching to unsuspecting youngsters the effective use of formal methods is one of the joys of life because it is so extremely rewarding. Within a few months, they find their way in a new world with a justified degree of confidence that is radically novel for them; within a few months, their concept of intellectual culture has acquired a radically novel dimension. To my taste and style that is what education is about. Universities should not be afraid of teaching radical novelties; on the contrary, it is their calling to welcome the opportunity to do so. Their willingness to do so is our main safeguard against dictatorships, be they of the proletariat, of the scientific establishment, or of the corporate elite." E. W. Dijkstra [2]*

In his "How do we tell truths that might hurt" [3] Dijkstra wrote that *"it is practically impossible to teach good programming to students that have had a prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration"* and that *"the use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offense."* Computer science faculty might consider such statements dramatic, insulting, even ridiculous, but I learned from a colleague that Dijkstra refused to accept students in his class if they had been exposed to BASIC or COBOL.
This, as well as his incorporation of "radical novelties" such as solving a problem without pencil or paper may help you understand the "cruelty" Dijkstra referred to in EWD1036.

Dijkstra was nothing if not consistent, holding himself to the same standards he held others.

He arrived at his most famous algorithm, known as The Shortest Path, in his head "while I had a cup of coffee with my wife on a sunny café terrace in Amsterdam," he has said.

> *"The algorithm for The Shortest Path was designed for the purpose of demonstrating the power of ARMAC at its official inauguration in 1956, the one for The Shortest Spanning Tree was designed to minimize the amount of copper in the backpanel wiring of the X1. In retrospect, it is revealing that I did not rush to publish these two algorithms: at that time, discrete algorithms had not yet acquired mathematical respectability, and there were no suitable journals. Eventually they were offered in 1959 to Numerische Mathematik in an effort of helping that new journal to establish itself. For many years, and in wide circles, The Shortest Path has been the main pillar for my name and fame, and then it is a strange thought that it was designed without pencil and paper, while I had a cup of coffee with my wife on a sunny café terrace in Amsterdam, only designed for a demo ..." E.W. Dijkstra [4]*

He also believed that solutions could – *should* – be elegant, and that elegance could prove elusive if a programmer's first step is to reach for a pen or pencil. (A 10 years earlier explanation of a future "slap":)

"I observed a few years ago that the moment at which mathematicians introduce avoidable complications very often coincides with the moment that they resort to such mechanical aids as pencil and paper," Dijkstra wrote in "A problem solved in my head." [5]"It was then that, for the sake of clarity of my own thinking, I decided to be less liberal with the use of pencil and paper and not to use them when I could avoid using them."

"ARROGANCE IN COMPUTER SCIENCE IS MEASURED IN NANO-DIJKSTRAS"

Alan Kay 1997

"Arrogance in computer science is measured in nano-Dijkstras," computer scientist and Turing Award winner Alan Kay said during his 1997 OOPSLA keynote. The quip, which produced a roar of laughter from the audience, and his ensuing criticism of Dijkstra is preserved on YouTube.

I listened carefully to the YouTube video, I found Kay to be far from eloquent. He began his keynote with an anecdote:

"He [Dijkstra] once wrote a paper of the kind that he liked to write a lot of which had the title 'On the fact that the Atlantic has two sides' [EWD611] and it was basically all about how different the approaches to computing science were in Europe, especially in Holland, and in the United States. In the U.S. here, we were not mathematical enough and, gee, in Holland, if you are a full professor you were actually appointed by the queen, and there were many other important distinctions made between the two cultures.

"So I wrote a rebuttal paper and it was called 'On the fact that most of the software was written on one side of the Atlantic,' and it was basically about – 'cause I have a math degree, too – that computers formed a new kind of math … they don't really fit well into classical math… It was about a kind of practical math. The balance was between making structures that were supposed to be consistent of a much larger kind than classical math had ever come close to dreaming of attempting, and having to deal with the exact same problems that classical math of any size has to deal with, which is being able to be convincing about covering all the cases." [6]

Defending "Arrogance"

Your Honor, ladies and gentlemen of the jury, Professor Dijkstra is accused of "arrogance."

Well, it is well know that at times, Professor Dijkstra expressed his strong opinions with critical irreverence, infuriatingly insensitive, but always with eloquence. To cite some extremes, he called the great logician Bertrand Russell a "dilettante" regarding his mathematical notation, accused John von Neumann of bringing (contra-productive) anthropomorphic terminology to computer science inspired by his work on the brain as "medieval speculations", and one of his favorite tirades against Software Engineering, coined "the Doomed Discipline," and even harshly, "How to program if you cannot." Even more extreme, he recommended that students in introductory programming courses should be prevented from the temptation to execute their programs, as they should be taught through mathematical logic to infer the correct program hand in hand with their proof of correctness.

Despite these points of view, I believe that some of these extremes were part of his dramatics, theatrical, and sometimes humorous avenues to deliver a forceful message for change, a poke in the eye for those asleep at the wheel, about the need to breakthrough deadlock, and the need to be bluntly honest. A model of "how to say truths that might hurt." A number of his critics address these extremes of his writings as if they are his entire position

Professor Dijkstra was indeed arrogant but about honesty, about the programming elegance, and about radical novelties in education. Even if extreme, I still prefer his Don Quixotesque exceedingly demanding goal of "logic proof" as science-base driving force, to the faith-based "every program has bugs" convenience. John von Neumann used to say that it is easier to explain science with god than without god.

I would argue that Professor Kay's and Professor Dijkstra's points of view are at the two at extremes; "not really fond of mathematics in programming" vis-à-vis of the de-empiricization of programming "craftsmanship" through mathematics towards the science of programming.

Although the two have pursued magnificent bodies of work that inspired many, they disagree in ways which have nothing to do with "truth." They each have been forcefully articulating their own philosophy and there is nothing wrong with that. On the contrary, philosophical discourse is a must when dealing with things as complex as Computing Science. Who wants to talk about the obviously neglected empiricism in specifying requirements of large codes, a really embarrassing subject? What is then a programmer to do?

In what follows, I will argue the Dijkstraian quest for programming de-empirization though mathematics. And to bring home the point I want to make about bringing philosophy out of the closet in computing science, I will go though an irreverent, and infuriating to some, tour of the principles of philosophy using the writing of the great mathematician Gian-Carlo Rota. The "axioms" formulation are my attempt to present Rota's argument and to show how relevant philosophy is to addressing the bottlenecks and failures in the software design of large systems, e.g., "Inevitability of failure" or the "Myth of precision" or the "Dictatorship of definitiveness."

I believe that if you choose to critique someone as eloquent as Dijkstra, you must at least strive to do so in a similar vein. It is probably unfair to ask someone to match Dijkstra's eloquence. Few can. But as Dijkstra offers this view about non-principle based criticism: "I love to be corrected. (Besides being a most instructive experience, being corrected shows that the other one cares about you.) If, however, they only get infuriated because I don't play my game according to their rules, I cannot resist the temptation to ignore their fury and to shrug my shoulders in the most polite manner. ... I have come to the conclusion that there are such things as 'disabling prejudices.'" [7]

In fairness, it is hard for me to believe that Professor Kay's piece recorded on Youtube, with its ramblings and profanity was part of a prepared text for his Keynote. Probably what happened was his delivery got emotional, and then inarticulate; "disabling prejudices" indeed! I see this as a sign of a deeper problem in computer science, called by some *The Software Crisis*.

Is Computer Programming an Empirical Science?

Reflecting on the nature of computing science, in general, and programming in particular, one needs to focus on the empirical aspects, mostly belonging to the software engineering focus of the discipline, as well as on the work towards the de-empirization of programming, via mathematical sciences.

John von Neumann talked about the de-empirization of the natural sciences with the exquisite clarity of his writing. He presented it by talking about the double face of mathematics. "The most vitally characteristic fact about mathematics is, in my opinion, its quite peculiar relationship to natural sciences, or, more generally, to any science which interprets experience on a higher than purely descriptive level." [8] He gave two such glorious examples: one being Geometry and the other Calculus which both started as natural, empirical sciences. Then its de-empirization happened by the mathematical method. "Some of the best inspirations of modern mathematics (I believe, the best ones) clearly originated in the natural sciences. The methods of mathematics pervade and dominate the "theoretical" divisions of natural sciences. In modern empirical sciences it has become more and more a major criterion of success whether they have become accessible to the mathematical method or to the near-mathematical methods of physics. Indeed, throughout the natural sciences an unbroken chain of successive pseudomorphoses, all of them pressing towards mathematics, and almost identified with the idea of scientific progress, has become more and more evident. Biology becomes increasingly pervaded by chemistry and physics, chemistry by the experimental and theoretical physics, and physics by the very mathematical forms of theoretical physics… One has to realize this duplicity, to accept it, and to assimilate it into one's thinking of the subject. This double face is the face of mathematics, and I do not believe that any simplified, Unitarian view of the thing is possible without sacrificing the essence."

Von Neumann's deep questions about the nature of intellectual work in mathematics can serve as a guide into our analysis of computing science: Is computing science an empirical science? Or, more precisely: Is computing science actually practiced in the way in which an empirical science is practiced? Or, more generally: What is the computing scientist's normal relationship to his subject? What are the criteria of success, or desirability? What influences, what considerations, control and direct his effort?

Dijkstra addressed these questions in his writings. In "Craftsman or Scientist?" [9] Dijkstra discusses the two extreme techniques in teaching programming. At one extreme is the "craftsmanship style," similar to the work of the future craftsman joining a master and "learning by osmosis, so to speak, the skills of the craft … a well-guarded secret." At the other extreme is the "scientist style." The future scientist learns from a teacher who formulates knowledge and skill as explicitly as possible through free interchange of knowledge and insights – "being non-secretive is one of their rules of professional conduct." A physician and a physicist, respectively, are examples of people who, more often than not, practice the two styles.

However, "mathematicians are somewhere in between: mathematical results are published and taught quite openly, but there is very little explicit teaching on how to do mathematics, and publishing besides the results also the heuristics that led to them is regarded by many as 'unscientific' and therefore, bad style: quite often the editor's censorship will try to prohibit their publication."

Dijkstra asks: *"Where along this scale should we place the teaching of programming?"*

Twenty-two years before Kay's 1997 comment, Dijkstra charmingly alluded to coming trouble: *"To make implicit knowledge explicit … we should realize that changing a craft into a science, and making public property of the secret knowledge of the guild, will always cause the guild members to feel threatened. For many a 'puzzle-minded' virtuoso coder of the early sixties, the [recent] scientific development … has been most unwelcome. … He feels like the medieval painter that could create a masterpiece whenever his experience enabled him to render proportion well, who suddenly found himself overtaken by all sorts of youngsters, pupils of Albrecht Dürer and the like, who had been taught the mathematical constructions that were guaranteed to surpass his most successful, but intuitive renderings. And with nostalgia he looks back to the good old days when his experience and feeling made him an outstanding craftsman. And we should realize that, as far as programming is concerned, the battle is still going on. – "craftsmen" [proposals] … had a pronounced anti-intellectualistic flavour: it stressed that students should be taught how to solve the problems of 'the real world' and that, therefore, the curriculum should pay as little attention as possible to 'abstract subjects.'"*

Dijkstra advocated a blending of the two teaching styles. The "disastrous blending, viz. that of the technology of the craftsman with the pretence of the scientist" is not the solution because "the craftsman has no conscious, formal grip on his subject matter, he just 'knows' how to use his tools. If this is combined with the scientist's approach of making one's knowledge explicit, he will describe what he knows explicitly, i.e. his tools, instead of describing how to use them! … It deserves a special warning because, besides being disastrous, it is so respectable!"

His preferred blending: *"As teachers of programming we should try to blend the technology of the scientist with the pretence of the craftsman."* Sticking to the technology of the scientist means being as explicit as we possibly can about as many aspects of our trade as we can. *"Now the teaching of programming comprises the teaching of facts – facts about systems, machines, programming languages etc. – and it is very easy to be explicit about them, but the trouble is that these facts represent about 10 percent, of what has to be taught: the remaining 90 percent is problem solving and how to avoid unmastered complexity, in short: it is the teaching of thinking, no more and no less."*

Computing Science and Philosophy

"Experimental psychology, neurophysiology, and computer science may turn out to be the best friends of traditional philosophy." Gian-Carlo Rota [10]

Perhaps the best way to explore the Dijkstra-Kay argument is to detour briefly into philosophy. You may think it is impractical and plays no role in computer science, but I would argue that the philosophy of computer science is at the heart of their debate.

We begin with the great Gian-Carlo Rota, whose writings matched the eloquence of Dijkstra. In his essay "The Pernicious Influence of Mathematics upon Philosophy," Rota showed us the parallels between the double life of mathematics and the double life of philosophy. (We will talk about the double life of computer science as well.)

Rota characterized the double life of mathematics as truth and proof. "In the first of its lives mathematics deals with facts, like any other science," he wrote. "The facts of today's mathematics are the springboard for the science of tomorrow." In its second life, Rota wrote, mathematics deals with proofs. "Every fact of mathematics must be ensconced in an axiomatic theory and formally proved if it is to be accepted as true."

In contrast, "In its first of its lives, philosophy sets itself the task of telling us how to look at the world. … Philosophical description make us aware of phenomena that lie at the other end of the spectrum of rationality that science will not and cannot deal with." Then "In its the second life, philosophy, like mathematics, relies of method of argumentation that seems to follow the rules of some logic."

But philosophy "has not been quite as comfortable with its double life," Rota wrote. In its first, philosophy "sets itself the task of telling us how to look at the world … making us aware of phenomena that lie at the other end of the spectrum of rationality that science will not and cannot deal with. The assertions of philosophy are less reliable than assertions of mathematics but they run deeper into the roots of our existence. Philosophical assertions of today will be the common sense of tomorrow."

Axiom 0: Goddess Reason

In its second life, "philosophy, like mathematics, relies on a method of argumentation that seems to follow the rules of some logic," but – unlike mathematics – the rules have "never been clearly agreed upon by philosophers, and much of the philosophical discussion since its Greek beginnings has been spent on method," Rota wrote. "Philosophy's relationship with Goddess Reason is closer to a forced cohabitation than to the romantic liaison which has always existed between Goddess Reason and mathematics."

Are we to believe that Professor Kay would issue a restraining order to keep Goddess Reason from darkening computer science's door?

Axiom 1: Philosophical disclosures are met with anger that we reserve for the betrayal of our family secrets

Rota: "Philosophical arguments are emotion-laden to a greater degree than mathematical arguments and written in a style more reminiscent of a shameful admission than of a dispassionate description. Behind every question of philosophy there lurks a gnarl of unacknowledged emotional cravings which act as a powerful motivation for conclusions in which reason plays at best a supporting role. To bring such hidden emotional cravings out into the open, as philosophers have felt their duty to do, is to ask for trouble. Philosophical disclosures are frequently met with anger that we reserve for the betrayal of our family secrets."

When Jonathan Edwards, a research fellow with the Software Design Group at MIT, was asked to contribute a chapter to Beautiful Code, published in 2007 by O'Reilly, he declined. *"Beauty is an idealistic fantasy," he later explained on the blog Alarming Development [11] "I hope that someday we will discover such principles. But in the meantime software design is still a matter of wisdom, not knowledge, and is therefore largely unteachable."*

He confided: "I am having trouble with this assignment. Telling an inspiring story about a beautiful design feels disingenuous. Yes, we all strive for beautiful code. But that is not what a talented young programmer needs to hear."

Then, as if betraying a family secret, he wrote: "I wish someone had instead warned me that programming is a desperate losing battle against the unconquerable complexity of code, and the treachery of requirements. I can't teach you how to design beautiful code, because I don't know how myself. I may have managed to get a few things almost right. … A lesson I have learned the hard way is that we aren't smart enough. … and above all to prize simplicity. Another lesson I have learned is to distrust beauty. It seems that infatuation

with a design inevitably leads to heartbreak, as overlooked ugly realities intrude. Love is blind, but computers aren't. A long-term relationship – maintaining a system for years – teaches one to appreciate more domestic virtues, such as straightforwardness and conventionality. Beauty is an idealistic fantasy: what really matters is the quality of the never-ending conversation between programmer and code, as each learns from and adapts to the other. Beauty is not a sufficient basis for a happy marriage."

Axiom 2: Dictatorship of Definitiveness

Rota: "Philosophers in this century have suffered more than ever from the dictatorship of definitiveness. The illusion of the final answer, what two thousand years of Western philosophy failed to accomplish."

Axiom 3: Inevitability of Failure

Rota: "A dispassionate look at the history of philosophy discloses two contradictory features: first, these problems [of philosophy] have in no way been solved, nor are they likely to be solved as long as philosophy survives; and second every philosopher who has ever worked on any of these problems has proposed his own 'definite solution,' which has invariantly been rejected by his successors. … Philosophers of the past have repeatedly stressed the essential role of failure in philosophy. The failure of philosophers to reach any kind of agreement does not make their writings any less relevant to the problems of our day. We reread with interest the mutually contradictory theories of mind that Plato, Aristotle, Kant and Comte have bequeathed to us, and find their opinions timely and enlightening, even in problems of artificial intelligence."

Axiom 4: The Myth of Precision

Rota: "The prejudice that a concept must be precisely defined in order to be meaningful, or that an argument must be precisely stated in order to make sense, is one of the most insidious of the twentieth century. … Looked from the vantage point of ordinary experience, the ideal of precision seems preposterous. Our everyday reasoning is not precise, yet it is effective. Nature itself, from the cosmos to the gene, is approximate and inaccurate. … The ideal of precision in philosophy has its roots in a misunderstanding of the notion of rigor."

We misunderstand the concepts of philosophy if we force them to be precise. One insightful metaphor due to Wittgenstein is that philosophical concepts are like the winding streets of an old city, which we must accept as they are, and which we must familiarize ourselves with by strolling through them while admiring their historical heritage. [12]

Axiom 5: Appeal to Psychology

Rota: " … to justify their neglect of most of the old and substantial question of philosophy [they argue] that many questions formerly thought to be philosophical are instead 'purely psychological." … Experimental psychology, neurophysiology, and computer science may turn out to be the best friends of traditional philosophy. The awesome complexities of the phenomena that are being studied in these sciences have convinced scientists (well in advance of philosophical establishment) that progress in science will depend on philosophical research in the most classical vein."

"And if I have to describe the influence PL/1 can have on its users, the closest metaphor that comes to my mind is that of a drug. I remember from a symposium on higher-level programming language a lecture given in defense of PL/1 by a man who described himself as one of its devoted users. But within a one-hour lecture in praise of PL/1 he managed to ask for the addition of about fifty new "features," little supposing that the main source of his problems could very well be that it contained already far too many "features." The speaker displayed all the depressing symptoms of addiction, reduced as he was to the state of mental stagnation in which he could only ask for more, more, more. … When FORTRAN has been called an infantile disorder, full PL/1, with its growth characteristics of a dangerous tumor, could turn out to be a fatal disease." – Dijkstra [13]

Axiom 6: The Illusion of Definitiveness

Rota: "The results of mathematics are definitive. No one will ever improve on a sorting algorithm which has been proved best possible. … Mathematics is forever. … The problems of philosophy are the least likely to have 'solutions.' ... The reality we live in is constituted by a myriad contradictions, which traditional philosophy has taken pains to describe with courageous realism. But contradiction cannot be confronted by minds who have put all their eggs in the basket of precision and definitiveness. The real world is filled with absences, absurdities, abnormalities, aberrances, abominations, abuses, with *Abgrund*."

A Critic's Dilemma

In my previous Conduit article about Dijkstra (part 1) [14], I derived (with a slightly different notation) a "criticism equation": $E=mc^2$ In the equation, the impressionistic quantities are: "*E*" is the "energy" of criticism, "*m*" is the "substance" of the critical message, and "*c*" the "authority" of the critic. Well, Kay's Turing award surely qualifies him for authority, but his *m* is so small that he got no *E* at all. There are two obvious desiderata for a conscious critic that creates a dilemma for her as they are somewhat in conflict. The first axiom, "non-demagogy," says that you should be critical only in areas where you have significant and recognized achievements. The second axiom, "non-personal," says that the critique is more effective when it stays away from the personal biases of the critic. The higher your achievement in an area, the tougher your criticism could be of lesser achievers, so you clearly are biased in your critique towards your kind, failing to satisfy the second axiom. The Spartan criticism of Dijkstra, hard to take by many, is at the root of the "disabling prejudices." He lived a Spartan life, holding himself first at the same high standards that he used to critique others' shortcomings.

"As a scientist Dijkstra was a model of honesty and integrity. Most of his publications were written by him alone. The few publications that he wrote jointly with his colleagues bear the unmistakable trait of his writing style. He never had a secretary and took care of all his correspondence alone. He never sought funds in the form of grants or consulting and never lent his name to the initiatives to which he would not contribute in a substantial way. 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. His supreme self-confidence went

together with a remarkably modest lifestyle, to the point of being spartan. His and his wife's house in Nuenen is simple, small and unassuming. He did not own a TV, a VCR or a mobile telephone, and did not go to the movies. In contrast, he played the piano remarkably well" [15].

As I put the final touches on this article, the recent announcement of Grigory Perelman's solution of the Poincare Conjecture marked another illustration that we must work on what we love. His is a victory of doers over talkers, a victory of the deep theory scientists over craftsmen. We should all celebrate Perelman's achievement and his receipt of the $1 million Clay Mathematical Millennium Prize.

"The question 'What is Mathematics?' is as unavoidable and as unanswerable as the question 'What is Life?' In actual fact I think it's almost the same question." Dijkstra [15]

Dijkstra's Axioms:

Axiom 0: Life = Mathematics

Axiom 1: Computer Science = Mathematics + Murphy's Law

Axiom 2: Beauty is Our Business

Axiom 3: Simplicity is a prerequisite for reliability

FEAR

In 1984, soon after my family fled to the United States, I began telling my friends about life in communist Romania – hard-to-believe stories from a weird world of dictatorship and limited freedom. I told them about the Carpathian president who forced people to attend rallies and chant his name. I described a government composed entirely of the Carpathian's family members; long lines at supermarkets whose shelves were barely stocked; mile-long lines of cars waiting a turn at the gas pump; spending the night in line so that the next morning we could claim our two-bottle ration of milk – available only to parents with young children.

This crazy dictator had a wife who, though only a lab technician in a pharmaceutical research institute, somehow managed to earn her Ph.D. in chemistry in just six months. What an achievement, the newspapers declared. She was immediately promoted to director of the pharmaceutical research institute, where she was listed as co-author – and first author – of the hundreds of papers published annually by the institute. But just in case this did not do justice to her leadership, her name was listed in bold-faced type that was double the size of the other authors.

How can one not promote such a talent to a position aligned well with her stature? It may come as no surprise that the dictator's wife was appointed Minister of Science. Newspapers celebrated the achievement with patriotic pride, and noted that no one had anything critical to say about the appointment, at least on the record.

Our new friends had difficulty believing our stories. Why was no one brave enough to stand up to the regime, they wondered? Such a thing could never happen here in the United States, they said. Could it?

It is hard to talk about fear. I wished for a metaphor, a story to offer my friends as a way to defend my seemingly fearful compatriots. We dealt with the daily problems by telling political jokes – oh, the safe haven of artistic ambiguity! – a folkloric form of prolific and creative protest that occasionally got some of us called up to a certain office where we were told that "on so and so day you told a joke about … we are concerned about you …"

One thing about dictators: Their time for justice comes. That Carpathian dictator and his wife were executed in the revolution. It is said that the members of the execution platoon could not restrain themselves when they marched to execute the couple; some starting shooting before reaching the wall.

They say that patriotism is the last refuge
To which a scoundrel clings
Steal a little an' they throw you in jail
Steal a lot an' they make you king
There's only one step down from here, babe
It's called the land of permanent bliss
What's a sweetheart like you doin'
in a dump like this.

Bob Dylan [16]

"Funding in genomics is measured in nano-Landers!" was a colleague's attempt at survival humor at a recent computational biology conference. The similarity with the nano-Dijkstras quote is only that. These two cannot be more opposite! I am afraid that just about now, the nano could become pico.

Afraid? I guess there are many types of fear. Fear of losing freedom scars you. It wasn't until 2002 that I recognized something similar in the United States. I was working at Celera Genomics, and from time to time we would receive in secret a message from a genomic scientist of stature – an apology for the actions of some of his colleagues. Clearly genomics people were afraid to say positive things in public about Celera for fear of losing their research freedom. I recognized this fear from the experiences in Romania that I was trying to forget. This time, though, I was in the communist republic of genomics.

Father-in-law vs. Pajamas

I sent a silly game of my own to Professor Dijkstra but it has never been published before. I am including it here asking for algorithmic solutions, and offer a prize for their optimal algorithmic solutions, but they have to be "in your head" solutions. The Prize, like the ones I use for my students solving the most difficult parts of the extra credit homework, a slice of Providence's famous, Pastiche Fruit Pie. Write to me at sorin@cs.brown.edu and I will publish winners in the next issue of the Lighthouses.

A Silly Game

A story. A young man lives with his father-in-law, a very active retired man. Among the duties of the father-in-law at home was to wash dirty laundry. It follows that he washes the son-in-law's pajamas too. The son-in-law's N clean pajamas, all different, are stored on a certain shelf S, in a white closet. They are arranged, as usual, in one stack as shown in Fig.1 stack S of pajamas Let B be the basket where the used pajamas are deposited in order to be cleaned. The son-in-law is a very absent-minded young man, and when he changes his pajamas, he acts as follows: throws the used pajamas in B, and puts on the pajamas from the top of S. As already mentioned, the father-in-law is very active, so he puts the laundry in the washing machine as soon as they occur in B. That is, till the washing moment no more than one pajamas has time to appear in B. After washing, the pajamas are returned immediately to the top of S.

After some time (say, years), at a moment when the son-in-law comes to change his pajamas, he discovers a very strange thing: the pajamas he is wearing and the one from the top of S are extremely worn out, while the rest of pajamas are almost new! Then he understood that all the time he has been wearing these two sets of pajamas. The explanation can be obtained as follows: while the son-in-law was wearing pajama 1 the father-in-law quickly washed pajama 2 and placed it on top of S. When he changed the pajamas, he took pajama 2 and put pajama 1 in B which ended up on top of S, and so on. The son-in-law has been wearing the sequence of pajamas 1,2,1,2,1,2 … Our first problem is how to avoid this unfair wearing of pajamas. Let us define a fair wearing of the N pajamas to be a sequence of N pajamas that is any permutation of them.

The Problem. Give an algorithm (if possible the simplest; solvable in your head) for a fair wearing of the pajamas. There is a caveat (inspired by the real-life situation): no communication between son-in-law and father-in-law should be required.

The story, continued. The son-in-law discovered the optimal algorithm for Problem 1. He started using it, so everything seemed to be okay. However, a happy event brought some changes. A son was born. Among the reorganizations involved in the house was the one concerning the clothes. Now, the son-in-law's pajamas were assigned to a small shelf. The pajamas were now arranged in several stacks, say M stacks of maximum height K.

Concerning washing, the father-in-law acts now… non-deterministically. He returns the washed pajamas to the top of any stack he wishes. The nondeterministic return proves troublesome. A new algorithm is needed for the problem. Find the two optimal algorithms for the two versions of the problem.

References

1. E.W. Dijkstra, EWD 1213: "Introducing a Course on Calculi" (1995)
2. E.W. Dijkstra, EWD 1036, "On the Cruelty of Really Teaching Computing Science" (1988)
3. E.W. Dijkstra, EWD 498, "How Do We Tell Truths that Might Hurt" (1975)
4. E.W. Dijkstra, EWD 1166, "From My Life (Written Because People Ask Me for These Data)" (1993)
5. E. W. Dijkstra, EWD 666, "A Problem Solved in My Head" (undated)
6. Transcript of YouTube video at http://www.youtube.com/watch?v=s7ROTJKkhuI
7. E.W. Dijkstra, EWD 1227, "A Somewhat Open Letter to David Dries" (1995)
8. John von Neumann, "The Mathematician," published in Works of the Mind Vol. 1, No. 1 (University of Chicago Press, Chicago, 1947)
9. E. W. Dijkstra, EWD 480, "Craftsman or Scientist?" (1975)
10. G.C. Rota, "The Pernicious Influence of Mathematics upon Philosophy" (1991)
11. From J. Edwards' blog, Alarming Development, http://alarmingdevelopment.org
12. E. W. Dijkstra, EWD 340, "The Humble Programmer" (1972)
13. S. Istrail, "Storytelling About Lighthouses: Criticizing Professor Dijkstra Considered Harmless," Conduit, v.17 no. 2. (2009)
14. K.R. Apt, "Edsger Wybe Dijkstra (1930–2002): A Portrait of a Genius," Formal Aspects of Computing (2002)
15. E.W. Dijkstra, EWD 720, "Why Correctness Must Be a Mathematical Concern" (1979)
16. B. Dylan, "Sweetheart Like You"