CSU-33012 – SOFTWARE ENGINEERING

BIOGRAPHY OF A SOFTWARE ENGINEER

Liam Ó Lionáird [19335530]

October 24, 2021

No living computer scientist has fascinated and enlightened more minds inside our discipline than **Donald E. Knuth**. His contributions to algorithm theory and analysis, and to relaying timeless knowledge through the mammoth *Art of Computer Programming*, are immeasurable and have laid the foundation for the last 50 years of computer science. Greater still, however, is his impact on the scholarly world at large; for nearly every modern academic paper of this century has been published on the back of a piece of software he designed—TEX. (It is also being used to format this essay.) Knuth's history as a software engineer is unique and perhaps understated, and this essay wishes to chronicle and examine it through his work.

To better understand Knuth's solitary, often strange path through software development, it is worth illustrating his computer science background. He first arrived on the scene in the early 1960s, after earning a PhD in mathematics. It is this mathematical background that drove his research, pioneering the *analysis of algorithms* to scientifically prove their effectiveness (introducing the use of big-O notation). His most defining work, *The Art of Computer Programming*, stresses the importance of mathematics as a foundation for all of computer science. In an early academic world where standards, languages, and machines were evolving all the time, he favoured designing his own systems based in rigorous logic, generic enough to be applied anywhere. Most famously, he fashioned a fantasy computer architecture—*MIX*—for illustrating concepts and exercises in *The Art of Computer Programming*. In his own words: "New languages go in and out of fashion... I am trying to emphasise concepts that are timeless."

Knuth's decision to create the TEX software arose from necessity while working on *The Art of Computer Programming* in the 1970s. Frustrated by the erratic quality of the physical typesetting used on his books, he set out to design his own digital typesetting system. He started work on it in 1978. His plans for the software prioritised extreme portability (TEX produces '.dvi' files—short for *device independent*—which can then be converted for printing) as well as extensibility (allowing people to alter TEX's code for their own purposes). It was also designed with mathematical typesetting in mind, including a special 'math mode' for inputting formulae as plain text.

The process of developing TEX took several years, including a complete rewrite in 1982, until it was finally 'frozen' with version 3 in 1989. (Minor updates have since added digits of π to the version number; currently it stands at 3.141592653.)

The length of its development likely owed to Knuth's painstaking design of every aspect from the ground up. The algorithm for spacing letters and words, for instance, was aggressively fine-tuned, with dozens of variables determining the 'glue' separating each glyph. Of particular challenge was the proper rendering of mathematical formulae, which Knuth had to standardise himself by examining various academic journals. To allow pixel-perfect font rendering (in an age before PostScript), Knuth designed his own font-description language called *METAFONT*, with which he designed an entire family of fonts for use with TEX. (Named *Computer Modern*, they remain standard fonts used in papers across all of academia, including this one.)

Most notably for TEX, Knuth also adopted a brand-new software programming paradigm—*literate programming*. He designed his own language called *WEB* for programming TEX, using this approach. Inspired by the idea of software as literature, TEX contains its own documentation *as* its source code, with every algorithm explained in plain, human-readable logic. (The source code of TEX is even available as its own self-generated book publication.) Knuth championed literate programming as a clearer, human-centric way to write software, abstracted from the computer and focused on the *thoughts* behind each design decision—in this way, poor decisions would be more obvious, and the design of software would be guided by the programmer's own stream-of-consciousness logic. "Instead of imagining that our main task is to instruct a computer what to do," he wrote in 1984, "let us concentrate rather on explaining to human beings what we want a computer to do."

For all the eccentricity of its design, TEX has remained an impressively stable and portable piece of software; able to run on nearly every system, and requiring very little maintenance over the years. Famously, Knuth has offered $2.56 (one 'hexadecimal dollar') to anyone able to find a bug in his software. (An update to TEX earlier this year—the first since 2014—included only five bug-fixes.) Knuth has released it as free software, and its source code to the public domain; thus enabling tools such as LATEX to build on top of it and expand its reach to millions of academics and publishers.

Knuth's unique approaches to software development has produced results appreciated by people around the world—but the approaches themselves have seen far less adoption. Many tools and philosophies adopted for TEX—its 'literate programming' approach, the *METAFONT* system—have mostly proven too esoteric to catch on outside the circle of TEX and its enthusiasts. (One notable exception is the literate-style 'notebook' programming environments of Jupyter and Mathematica.) Knuth himself has acknowledged this, admitting that his methods expect people to be as good at writing and design (and at adopting entirely new concepts) as they are at programming. These methods have shown more age as decades pass and software standards solidify, and the impact of Knuth's design philosophy will likely only further dwindle. Yet still, it represents a curious far-away world of possibility—as inspiring as it is alien—carved out by one man daring and capable enough to reinvent the art of programming on his own terms.