The Art of Teaching Computer Science: Niklaus Wirth

Charles Severance, *University of Michigan*

With a goal of improving how computer science is taught, Niklaus Wirth created some of the field's most influential programming languages, including Pascal, Modula, and Oberon.

At some point in the 1960s and 1970s, computer scientists transitioned from focusing all their energy on building hardware at universities to purchasing commercial hardware and learning how to write software for those computers. The field's center of gravity moved from mathematics and electrical engineering to the emergent software-oriented CS.

One of the first issues in CS was how to design easy-to-learn programming languages that produce efficient, reliable, and easy-to-maintain code. The result of infinitely varying requirements for programming languages led to the development and use of literally thousands of distinct computer languages over the past 50 years.

I recently spoke with Niklaus Wirth about programming languages; you can view the full interview at www.computer.org/computingconversations.

ALGOL TO PASCAL

Wirth earned a PhD in electrical engineering and computer science from the University of California, Berkeley, in 1962 and worked at Stanford University as an assistant professor from 1963 to 1967. In 1968, he returned to Switzerland, where officials at the University of Zurich and ETH Zurich asked him to found a CS department:

> When I returned to Switzerland, I got an invitation from the University of Zurich and the ETH to introduce computer science as a new subject. Being a software man, I looked at what tools were available, which was rather a disappointment. There was Algol, which had recursion and good structure, but it had a lousy implementation, and I felt that using Algol could only reduce the chances of it [CS] ever being accepted. On the other side, there was only Fortran, which I found unsuitable for teaching, as was the assembly code.

Wirth designed a language that he felt improved on Algol60 and was also suitable for teaching; he proposed adopting his new language as Algol68:

> I decided to continue my work from Stanford and implement not another Algol compiler but what later became known as Pascal. I had been a member of the IFIP [International Federation for Information Processing] working group [2.1] working on a successor to Algol60, and there were finally two proposals, one from Aad van Wijngaarden from Amsterdam and one from me. You might say that I had lost out. I decided to implement it in spite of it all because I needed it for teaching, which led to Pascal in 1969. In 1971, I used it for the first time in an introductory programming course.

In the early days of CS, academics readily shared their innovations with each other, often shipping nine-track tapes with source code back and forth. Wirth helped with efforts to port the Pascal compiler to other hardware architectures. Pascal was widely available on several different mainframe computers when the microprocessor revolution started:

> The real breakthrough came with the advent of the microcomputer, like the Apple II, Tandy, and others. They brought out UCSD Pascal, and there was a Pascal implementation from Borland called Turbo Pascal. They were selling not only compilers but also an integrated system with a text editor and debugger for something like $50, and that really made the difference at a time when compilers still cost thousands of dollars for large machines. [People] started learning

 Published by the IEEE Computer Society

programming from scratch, and that is how computing was brought into homes and schools.

Because Pascal was a structured language, it was far more suitable for building production-quality software on these new microcomputers. With strong type-checking and well-defined interfaces, it was superior to Basic for anything other than the smallest applications.

BEYOND PASCAL

Because Wirth distributed Pascal freely and didn't charge for a license, companies such as Borland could ship very low-cost products, starting with his code:

> Pascal was a public good. I had very little interaction [with Apple, Tandy, or Borland]—really, almost none. The Atlantic Ocean is too wide, or at least too wide for close interaction. We always distributed our software for free, covering the cost of the tape, so nobody had an obligation to fall back on me.

As Pascal experienced success around the world in the 1970s, Wirth remained focused on building the right languages and environments that would be ideal for teaching CS. From 1976 to 1977, he took a sabbatical at Xerox PARC and started thinking about a more modular version of the Pascal language that developers could use to build larger programs. He chose to name the follow-on language Modula to highlight the language's modular aspects:

> I didn't name my other languages Pascal, and from a commercial point of view, that's regrettable, because if I had called Modula "Pascal-2" and Oberon "Pascal-3," they would have had better success. Modula-II came nine years after Pascal. It was a language designed for system development, influenced also by MESA, which was developed at Xerox PARC, where I spent a sabbatical. MESA itself was based on Pascal. The primary new feature of Modula was the module, and with it, the interface specification. Separate compilability with tight interface type-checking. That's of course what was missing with Fortran, so linking different Modula modules together is as safe as programming in one module, and that was an absolutely crucial thing.

Because Pascal was a structured language, it was far more suitable for building production-quality software on these new microcomputers.

Although not naming Modula "Pascal-2" lessened its commercial uptake, it also had the effect that Wirth could continue to focus on developing technology to better teach CS. During his sabbatical at PARC, he also became interested in hardware:

> At Xerox PARC, I was given an Alto computer for myself alone, on my desk, and that was an absolute change in the way computers were used. At home [in Switzerland], I still had a terminal linked with a thin wire to a big machine that was shared with hundreds of others. I decided I didn't want to program [at home] with those old dinosaurs anymore, and I had to have one of these things, too, but they weren't for sale—they could not be bought.
>
> The only thing I could do was build one myself, and that's how I diversified into hardware. Fortunately, I had been trained as an electrical engineer, so it was a bit easier. But in something like 15 years, electronics had undergone a big change. I was trained on vacuum tubes, and now there were not only transistors but also integrated chips. It was really fascinating; with very little money, we built a little workshop and some prototypes. Modula and the compilers and the operating systems were closely connected to the Lilith computer, as it was called.

In 1987, Wirth took a second sabbatical at PARC and used the time to think about building an object-oriented version of Pascal/Modula that he would call Oberon:

> I and my colleague Jürg Gutknecht became convinced that the future lay, particularly for teaching programming, in simpler languages. Modula had become overloaded with features and facilities, so we wanted to clean it up, which resulted in Oberon. We essentially added only one feature—an important one—and that was type extension. Together with procedure variables, which were already present in Modula, you could implement the full scheme of object orientation.

Just like with the Lillith computer 10 years earlier, Wirth and his graduate students also built a workstation for Oberon. The Ceres was designed from the ground up to support the Oberon language and Operon operating system. Operon was a bit-mapped windowed operating system, and tools like word processors were developed in Oberon.

FPGAs

In the 1990s, Wirth turned his focus to field-programmable gate arrays (FPGAs) to explore hardware design. His teams built languages to program FPGAs and made increasingly sophisticated use of these hardware devices.

His interest in using FPGAs led to the project that Wirth continues to work on to this day. After retirement in 1999, Wirth progressed his work with FPGAs to the point where he has fully specified a RISC CPU that can be implemented using an FPGA (www.inf.ethz.ch/personal/wirth/Articles/FPGA-relatedWork/RISC.pdf). His current task is to alter his Operon operating system so that it can run on

his FPGA-based RISC computer. His plan is to build a complete computer using a $100 Xilinx FPGA board:

> I felt that we should apply the same principles of simplicity that we used in software to hardware, and this is now possible because of FPGAs. I have a Xilinx development board with an FPGA on it. I implemented a microprocessor. I call it RISC, but it's much simpler than the ARM, MIPS, or SPARC, again concentrating on what is essential and presentable for students. The processor's code takes about three pages. I wrote an Oberon compiler for that RISC architecture that includes a linker and downloader.

Wirth jokingly says that "retirement is the best form of tenure because you don't need to go to committee meetings." His current goal is to finish his FPGA-based RISC processor, release his updated Oberon environment for that processor, and revise his Oberon book to include chapters describing the RISC processors and its hardware design.

When Wirth's hardware, software, and book are complete, it will hypothetically be possible to build a lab full of Oberon workstations in which, as part of their evolving CS education, students could start from writing a "hello world" application in Oberon, move on to understanding the details of object-oriented programming and operating system design, and then into computer architecture and hardware design. All the while, the book they'll be learning from will describe every layer and abstraction of the hardware and software in front of them. Everything will be as simple, elegant, understandable, and easy to learn as it can be so that students can learn as much as possible as quickly as possible. Niklaus Wirth would have it no other way.

***Charles Severance**, Computing Conversations column editor and* Computer*'s multimedia editor, is a clinical associate professor and teaches in the School of Information at the University of Michigan. You can follow him on Twitter @drchuck or contact him at csev@umich.edu.*