

Everything My Computer Science Teacher Taught Me Was Wrong

*Radical changes in hardware and software are transforming what you'll be doing with technology
Microsoft's Andrew Herbert explains how.*

In the 20th century, computer software was designed to overcome the many limitations of computer hardware of the time. As we enter the 21st century, computer resources have become so abundant that the new challenge for software designers and engineers is how to fully exploit them.

Scientists at Cambridge University in the UK did much of the early computing research, helping to define many of the concepts that underpin the ways we went about programming computers. My colleagues and I at the Microsoft Research Cambridge Lab continue building on the ideas from those computer pioneers. But while many of the foundations they laid underpin our research, we've had to question a number of their assumptions.

► The legacy of Moore's Law

In 1965, when Gordon Moore was director of research at Fairchild Semiconductor Corp., he defined what is now known as Moore's Law. It's the empirical observation that the number of transistors in a given area on an integrated circuit doubles approximately every 24 months.

Moore's Law has defined the way in which the personal computer industry has grown; indeed, companies like Microsoft have grown on the back of the increase in computing power. In founding Microsoft, Bill Gates and Paul Allen had the vision to see how the microprocessor would take on the responsibilities that previously had been given to mid- and mainframe computers, to see that computing would become much cheaper as computers got smaller. They realized that smaller and more powerful computers would become part of everyday life.

One of the most dramatic impacts of Moore's Law has been on the cost of computing. When I was an undergraduate, the university mainframe computer cost several millions of pounds to purchase and filled a whole room. As a research student, a UNIX computer cost several tens of thousands of pounds and the operating system costing thousands of pounds more. Compare that to the few hundred pounds for a modern PC that comes with software preinstalled.

We've now reached a stage with Moore's Law that we have, on an individual chip, more transistors than we know what to do with, making the challenge of the 21st century too many transistors and not enough ideas. One problem is that while we can still pack ever more transistors in a given area, we cannot "clock them" or run them much faster. So we are now using these transistors to build independent processors that can run in parallel in a single package called a multi-core processor.

► The single threaded program

That brings us to the first 20th century idea that's become obsolete: the idea of a computer program as a single sequential series of instructions called a "thread". Going back to 1952, on an early computer such as the Cambridge University EDSAC 2, programs were read in on paper tape — a sequential flow of instructions doing one single thing after another. It was easy to understand what the computer was doing, because at any point, only one thing was going on.

To fully exploit a new generation of multi-core processors, we need to feed parallel sets of instructions to the machine. This makes software programs much more complex. As one result, we've had to invent new programming languages that make it easier for us to express parallel computations. But we also need more programming tools to help us design parallel software. And we need better debugging tools that make it easier for us to diagnose problems. This is a big challenge and it's going to change the way we teach programming as well as change the things a programmer is going to need to know.

► Low-level programming languages

Going the way of the now-obsolete idea of sequential programming, are programming languages that allowed the programmer to issue detailed low level hardware instructions in a quest to make things efficient. The cost of "efficiency" was often reliability. For example, many of those languages allowed you to use memory that could not be guaranteed to belong to the program as the programmer could directly issue memory access instructions. It was easy to lose track of what was going on and make all kinds of low-level, simple errors. And that was the cause of many of the failures that we saw in the past.

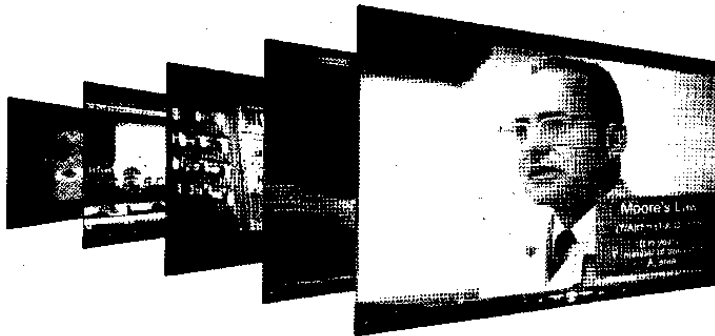
There are now new languages that make it impossible for programmers to make those kinds of mistakes. They better capture the ideas within the applications programmers are trying to build rather than the details of the computer instructions. Programmers rely on compiler software to translate those high-level programming ideas into the basic language the machine understands. Once considered an extravagant use of memory, compilers are now essential tools. We are moving towards designing yet higher level languages with greater levels of automation and self-checking that eliminate programming mistakes.

► Screens on desks

Today's screens make it uncomfortable to read anything of any real length, so users prefer to print things out on paper. What happens if you can have an electronic screen that mimics a piece of paper; that you can take to the beach, for example, and read in bright sunlight? The technologies for those types of products with enough memory

The top-five obsolete software ideas

- 1 The single threaded program
- 2 Low-level programming languages
- 3 Screens on desks
- 4 Virtual memory
- 5 Hierarchical file systems



to hold volumes of books are now being realized at reasonable costs. One example is the appearance of displays based on thin plastic films. They are very cheap and easy to manufacture, and they get us away from the idea that a display has to be a screen in a box that we sit in front of.

Researchers are exploring methods for turning just about any surface into a screen. If we couple the developments in display technologies with some of the developments in image processing to help us with image recognition, the technology could be extended to recognize gestures or things that are standing in front of or on top of a display. This opens a whole new way of interacting with computers. By putting sensors behind a display, for instance, you can interact directly with the screen, manipulating images with your fingers, spinning things around and stretching them with simple gestures. With technologies such as multi-touch interaction, we are just beginning to explore the possibilities. You can see how the idea of sitting in front of a screen and using a keyboard to interact with a computer will become obsolete.

► Virtual memory

Since 1970, we've seen the size of memories grow about a million fold. Indeed, I can remember the University of Cambridge buying a megabyte of memory for its IBM mainframe at about £2 million (US\$4 million), whereas now you can buy a gigabyte of memory on a small flash memory stick for £50-100 (US\$100-\$200).

The Cambridge Cap Computer, dating from 1975, was built to explore the idea of virtual memory. The concept was to run more programs for more users by swapping them between the main memory of the machine and a computer disk. That was called dynamic virtual memory. It became a very important idea, but with the advent of larger and cheaper main memories it's become obsolete.

Within a few years we will probably be able to carry a terabyte of personal storage, enough to hold all of the audio and video you'd want to use in a lifetime. This presents new challenges: how do we organize all of this information and how do we search through it?

► Hierarchical File Systems

The next obsolete software idea is that of finding files by putting them into some kind of hierarchical system of named folders. We

now have too much stuff for this to work. Modern database technology will be used to store and search through the terabytes of data that we will need to store. Image recognition techniques will be used to process and auto-collate the thousands of pictures and videos that we will generate.

Computer hardware resources have grown exponentially in the 20th century and this pace is expected to continue into the 21st century. There are now numerous new opportunities and challenges in developing the software that runs on that hardware. We have to learn how to manage and exploit concurrency, we have to adapt to handling large amounts of data, and we have to think about new forms of interaction with a computer. If we can change the way we think about software in the light of these developments, we have the opportunity to realize the true potential of software in the 21st century.

Watch the video

Andrew Herbert demonstrates how software is changing in the 21st Century on *TechNet Spotlight*, part of the British Airways in-flight entertainment system on selected routes.

Microsoft resources for IT Professionals at microsoft.com/technet

- Live events & webcasts
- Evaluation Software and Betas
- Community & Forums
- Troubleshooting & Virtual Labs
- IT Training & Certification