On the Nature of Computing

Computing is its own virtual world, bound only by its practitioners' imaginations and creativity.

I would like to propose that computing's innate agenda is the virtual, rather than the natural or the artificial.

Each of us in the computing community experiences periodic bouts of navel gazing about the nature of our business. The related public debate typically polarizes us along a spectrum between engineering and science. At the engineering end are usually the engineers who design and manage systems, networks, and operating systems; at the science end are the ideas describing computability, complexity, and information theory. An extreme view of each end places practitioners within university electrical engineering departments, and theoreticians within university mathematics departments.

I studied the natural sciences at Cambridge University as an undergraduate. I was taught the value of studying the natural world, along with the use (and advance) of mathematics to describe and understand (and predict) its behavior. I have also spent more than a decade teaching courses in an electrical engineering department, where artificial systems are built according to models (often mathematical) with reliable and predictable behavior. Computing has never established a simple connection between the natural and the mathematical. Nowhere is this lack of a clear-cut connection clearer than when Ph.D. students select a problem for their thesis work; their dilemma is the key to understanding why computing represents a third place in the world of discourse—distinct from the natural and from the artificial of science and engineering.

Computing involves (virtual) systems that may never exist, either in nature or through human creation. Ph.D. students find it difficult to settle on a topic because the possibilities are endless and the topic may have no intersection with the real world, either in understanding a phenomenon or in creating an artifact. In trying to define the nature of computing I completely disagree with the late Nobel Prize physicist Richard Feynman.[1] Computing often results in a model of something. Although an object or process that interacts with or describes the real world may be the outcome, it does not have to be.[2]

Computing's disconnection from physical reality has an important consequence when explaining to the public what it is computer scientists do, whether to schoolchildren, noncomputing users in general, or funding agencies and decision makers. Unlike the artificial (the engineering end of the spectrum), some of what we do may not be obviously useful and therefore attractive to commerce and governments for optimizing social welfare or profit. Unlike the natural world (the scientific end of the spec-

[1] "Computer science also differs from physics in that it is not actually a science. It does not study natural objects. Neither is it, as you might think, mathematics; although it does use mathematical reasoning pretty extensively. Rather, computer science is like engineering; it is all about getting something to do something, rather than just dealing with abstractions, as in the pre-Smith geology." Richard Feynman, from the book *Feynman Lectures on Computation* (1970).

[2] I am tempted to lay claim to the term "magic" [1]. A lot of what computer scientists do is now seen by the lay public as magical. Programmers (especially systems engineers) are often referred to as gurus, sorcerers, and wizards. Given the lofty goals of white magic, understanding the power of names and the value of pure thought, the power of labels is indeed attractive. However, many historically compelling reasons argue against this connotation, including the sad history of Isaac Newton's alchemical pursuit of the philosopher's stone and eternal life, and the religiously driven 17th century witch trials in Salem, MA, and other seemingly rational explanations for irrational behaviors.

LISA HANEY

trum), some of what we do may not necessarily be "for the advancement of pure knowledge" and therefore a priori worthwhile. In some sense, though, what we do underpins both of these engineering and scientific activities.

I am comfortable endorsing the claim that computing is less worldly than, say, cosmology. On the other hand, due to the possible use of computing as part of the foundation of practically any kind of system—whether physical or abstract—anyone is likely to build today, computer scientists can also claim that computing is inherently more useful than engineering.

Examples of the Virtual

To illustrate my argument, consider the following examples of the virtual I've selected from the history of computer science:

Virtualization. Within the discipline of computer science itself, the concept of virtualization represents a first-class tool. When confronted with intransigent engineering limitations of memory, processors, I/O, and networks, we've commonly taken the abstract approach. For example, we create virtual memory systems to replace one piece of hardware with another as needed to overcome capacity/performance problems and to choose when it's appropriate to do so; we replace inconvenient low-level processor interfaces (the instruction set) with virtual machines (such as VM, vmware, Xen, and Denali), to provide a more convenient (and stable) interface for systems programmers. We might provide a single API to all I/O devices, so programs need not worry whether, say, an MP3 file is being loaded from a tape, a magnetic disk, an optical disc, flash RAM, or even networked media. We also might replace a network with a virtual private network, allowing users to behave as if they were in an Internet of their own.

Virtual communities. In the emerging world of grid computing (notably in the U.K.'s e-Science program), we are creating virtual communities of scientists with virtual laboratories and computing resources dedicated to supporting "in silico" experiments, replacing the expensive, error-prone "in vivo" or "in vitro" experiments of the past. Here, we have virtualized natural systems, whether they involve fluids (such as the atmosphere, oceans, and plasma) or complex biological systems (such as genomes, proteins, and even whole ecologies).

Entertainment. The convergence of computer games and the movie industry represents the clearest evidence to support my view that computing is a wholly new discipline. The world of entertainment imposes no natural or artificial constraints on what a system may do. The only limit is the imagination of its creators, combined with knowledge and skills from the computing discipline. Constraints may be imposed from the discipline itself (such as computability, complexity, and plain affability) but may often be orthogonal to the goals (if any) of the computation.

Historically, simple examples of virtual worlds have been used in both games and online environments, as well as for playing with alternate realities (such as in artificial life), so this view is not something that has suddenly become true. It has always been one of the exciting but difficult aspects of working in computing that the bounds are not set from outside the field but by our own choice of what research projects we most want to work on and see developed.

Conclusion

Occupying a third place in human intellectual culture, computing is not bound by the need to describe what does exist (as in natural science) or what can be built in the real world (as in engineering). This place is the virtual. Although we computer scientists do not need to be complete, consistent, or correct, we have the tools to choose to be part of these categories whenever we wish our systems to be complete, consistent, or correct. C

Reference

1. Penrose, R. *The Road to Reality: A Complete Guide to the Laws of the Universe.* Jonathan Cape, London, U.K., 2004.

Jon Crowcroft (Jon.Crowcroft@cl.cam.ac.uk) is the Marconi Professor of Communications Systems in the Computer Laboratory at the University of Cambridge, Cambridge, U.K.