DOI:10.1145/2347736.2347737

Vinton G. Cerf

Where is the Science in Computer Science?

We are all members of the Association for Computing Machinery. Sounds sort of electromechanical doesn't it? Given today's computing technology, it is probably a good

thing we are mostly known as ACM! There was a time when the physical artifact—the computer—really was the focus of attention. These behemoths occupied rooms full of equipment. Now, in fairness, if you have ever visited a cloud computing data center, the dominant impression is still a (vast) room full of machinery. But we carry huge quantities of computing power in our pockets and purses too. Computing is a remarkable artifact and its origins centered on the ability to make a piece of equipment calculate under programmable control. Alan Turing, whose 100th birthday we celebrated this year, drew dramatic attention to the artificiality of these systems with what we now call the Universal Turing Machine. This conceptual artifact emphasizes the artificial nature of computation.

In the physical world, science is largely about models, measurement, predictions, and validation. Our ability to predict likely outcomes based on models is fundamental to the most central notions of the scientific method. The term "computer science" raises expectations, at least to my mind, of an ability to define models and to make predictions about the behavior of computers and computing systems. I think we have a fairly good capability to measure and predict the physical performance of our computing devices. We can measure clock speeds, latencies, memory sizes, and computational capacity against standard computing tasks. In my view, however, we are much less able to make models and predictions about the behavior and performance of the artifact we label "software." An almost flippant analogy is the difference between measuring, modeling, and predicting neural brain functions and trying to do the same for "thought."

That software is an artifact seems obvious. Moreover, it is a strikingly complex artifact filled with layer upon layer of components that exhibit dependencies and complex and often unpredicted (not to say unpredictable) behaviors. Even though we *design* software systems and ought to have some clues about how these systems behave and perform, we generally do not have a reliable ability to anticipate the states these systems can get into, their vulnerabilities, their performance, and ability to adapt to changing conditions.

When we write a piece of software, do we have the ability to predict how many mistakes we have made (that is, bugs)? Do we know how long it will take to find and fix them? Do we know how many new bugs our fixes will create? Can we say anything concrete about vulnerability? What about the probability of exploitation? Murphy's Law suggests that if there is a bug that can be exploited for nefarious purposes, it will be. ACM Turing Award recipient Fred Brooks' wonderful book, *The Mythical Man-Month*[1] captures some of the weakness of our understanding of the nature of software. A complementary look at this topic is found in ACM Turing recipient Herbert A. Simon's *The Sciences of the Artificial*.[2] Chapter 8 deals with hierarchy and complexity, touching on the way in which we try to bound complexity through modular and hierarchical structures but are still challenged by the emergent behaviors masked, in some ways, by the beguiling apparent simplicity of the hierarchy.

The richness of our field has only grown in the 65 years of our existence as an organization. Computers, computing, software, and systems are seemingly omnipresent. We are growing increasingly dependent upon what must be billions of lines of code. Some unknown wag once quipped that the only reason all the computers in the world have not failed at once is that they are not yet all on the Internet. But that may be coming (not the collapse, I hope, but the interconnection of a vast number of programmable devices through the Internet or its successor(s)).

As a group of professionals devoted to the evolution, understanding, and application of software and hardware to the myriad problems, opportunities, and activities of modern society, we have a responsibility to pursue the science in computer science. We must develop better tools and much deeper understanding of the systems we invent and a far greater ability to make predictions about the behavior of these complex, connected, and interacting systems. I consider membership in the ACM a mark of recognition of that responsibility. I hope you share that view and will encourage others in our profession to join ACM in the quest for the science in our discipline. C

References

1. Brooks, F.P. *The Mythical Man-Month*, Anniversary edition, 1995. Addison-Wesley, Reading, PA, ISBN 0-201-83595-9.
2. Simon, H.A. *The Sciences of the Artificial*, 3rd edition, 1996. MIT Press, Cambridge, MA, ISBN 0-262-19374-4.

Vinton G. Cerf is Vice President and Chief Internet Evangelist at Google Inc. and the president of ACM.