Dr. Wm. A. Wulf is on leave from the University of Virginia, where he is a University Professor and AT&T Professor of Engineering in the Computer Science Department, to serve as President of the National Academy of Engineering (NAE). Together with its sibling, the National Academy of Sciences, the NAE is both an honorific organization and an independent, authoritative advisor to the government on issues involving science and technology.

Prior to joining the University of Virginia, Dr. Wulf was an Assistant Director of the National Science Foundation, responsible for computing research, the national supercomputer centers, and the NSFnet (predecessor to the Internet as we know it now). Prior to NSF, Dr. Wulf founded and was CEO of Tartan Laboratories, a software company in Pittsburgh. Tartan was based on research Dr. Wulf did while on the faculty of Carnegie-Mellon University.

Dr. Wulf holds a BS in Engineering Physics and an MS in Electrical Engineering from the University of Illinois and a PhD. in Computer Science from the University of Virginia. He has conducted research in computer architecture, programming languages, optimizing compilers, and computer security. He is a Fellow of the IEEE, ACM, AAAS, AWIS, and the American Academy of Arts and Sciences, and is a member of the National Academy of Engineering.

Wulf & Jones
Quill Spring
3897 Free Union Road
Charlottesville, Va. 22901

5/14/02

William Wulf
National Academy of Engineering

I would like to suggest several, perhaps related, challenges. They all deal with how we handle *very* complex (computing) systems.

Challenge 1: How can we construct software such that a single individual can have "complete intellectual control" of a billion-line program?

I would presume that such a system would not have a billion lines of code, of course, so the reference to a billion lines is only intended as a descriptor of a program of such complexity that if we were to build it today it would require a billion lines of code.

By "complete intellectual control" I mean that the individual completely understands the design and operation of this program, can predict all of its behaviors, can make a convincing argument for its "correctness," and with some facility can modify it to behave differently.

Challenge 2: All of the interesting and useful behaviors of biological systems are emergent. Many, if not most, of the undesirable properties of programs are emergent. The challenge is to understand the nature of emergent properties sufficiently well that we can make the emergent properties of software interesting and useful rather than undesirable.

The useful properties of biological systems are the result of billions of years of natural selection; the question is whether we can develop the theory, mathematics and engineering process that achieves the same effect on human time scales.

Challenge 3: Make discrete mathematics into a useful tool!

I think it's damning that the formal specification of a program is about the same size as the program itself. Moreover, such specifications cannot be analyzed to tell us anything useful about the resulting program.

I suspect a major difference between logic and the calculus is that the latter was developed in parallel with physics while the former had no driving application other than mathematics itself. The fact that it's possible to specify a program's behavior in the predicate calculus demonstrably does not mean that it's easy or that the result is particularly useful.

Challenge 4: What's the analog of "continuity" in the digital domain?

A lot of the power of the calculus comes from the observation that small changes in the input of a function result in small changes in the output. No such thing is assumed to apply in digital systems.

The primary memory of my laptop can be in any one of $1^{10000000000000000000}$ states; in principle, a one-bit change can result in taking one to a radically different, unrelated state, in principle! In practice, probably not. Yet, the need to presume this lack of a digital “continuity” has a lot to do with the size and complexity of specifications. Is there a way to build systems that would lead to systems in which some analog of “continuity” made a new mathematics of specification and analysis possible?