"I’m Not a Computer Scientist, but ..."

David Alan Grier, George Washington University

Computing is opaque to some scientists and science is opaque to some software engineers.

Graduate students can form communities in an instant. When they gather as strangers—at a conference, class, or party—they often demonstrate an ease in bonding with and trusting others. After exchanging bits of information, they quickly find that they have a common background and experience. “Your ways are my ways,” they tell each other. “Your concerns are my concerns.”

I saw such a community flourish at a recent summer school program as the participants discussed the impact of computer simulation on scientific research. For a week, 15 students laughed and smiled and gossiped while they discussed the ideas of theoretical physicist Fritz Rohrlich, who argued 25 years ago that we were “at the threshold of an era of new scientific methodology.” He claimed that simulation was a “strength of theoretical mathematics while avoiding its associated weakness.” However, most of the students ignored the strengths of simulation and focused on its weakness, arguing that simulation was “epistemologically opaque”—a term coined by philosopher Paul Humphreys. To a philosopher, simulation is epistemologically opaque because it can’t be fully verified by a scientist. A scientist can verify that a simulation produces the same data that’s produced by nature, but can’t verify that the simulation produces those numbers the same way that nature does.

Each speaker offered a different reason for epistemological opacity, including the complexity of software, the division of programming labor, the use of software libraries, and even the age of many simulation languages. The most novel idea was based on a phrase that one student, Katerena, heard repeatedly while studying programming practices on a large simulation project: “I’m not a computer scientist, but ...”

Katerena observed that the phrase was being used to justify bad programming practices and to hide the code’s internal operation. She said that her subjects, the researchers on the project, had learned programming from each other and hence were mediocre at it; they would recommend unsuitable computer languages, accept technical debt, code inefficient algorithms, design faulty file structures, or cling to awkward factorizations. In almost every case, the bad habit was introduced with the phrase in question.

From Katerena’s research, it was a little too easy to conclude that scientists need better software engineering training. The “I’m not a computer scientist, but ...” phrase not only tells us that computing is opaque to some scientists, but also reminds us that science is opaque to some software engineers. Software engineering permeates scientific research (one speaker characterized software engineering as “contaminating science”). For most projects, scientists need software to collect, simulate, and analyze data. More science is done on the desktop than in the lab or field.

Software engineering has the goal of making things that work, and scientific research aims to understand what’s true, even when it can’t make things that work. To one, a simulation is a program that has to work; to the other, it’s a way of communicating the ideas we believe to be true. They have enough in common to be part of the same community, but they’ll never be the same thing.

DAVID ALAN GRIER is an associate professor at George Washington University. Contact him at grier@gwu.edu.