Vigorous Public Debates in Academic Computer Science

JOHN REGEHR

**EXPERT-CURATED
GUIDES TO
THE BEST OF
CS RESEARCH**

Research for Practice combines the resources of the ACM Digital Library, the largest collection of computer science research in the world, with the expertise of the ACM membership. In every RfP column experts share a short curated selection of papers on a concentrated, practically oriented topic.

This installment of Research for Practice features a special curated selection from John Regehr, who takes us on a tour of great debates in academic computer science research. In case you thought flame wars were reserved for Usenet mailing lists and Twitter, think again: the academic literature is full of dramatic, spectacular, and vigorous debates spanning file systems, operating system kernel design, and formal verification. Please enjoy!

– Peter Bailis

Science is often portrayed as a world of cold logic and hard facts, but as human beings we have great difficulty keeping emotion and prejudice out of the picture—whether the issue at hand is inconsequential (“Is Pluto a planet?”) or threatening to our existence (“Can we stop global warming?”). Historically, as Galileo’s encounters with the Roman Inquisition showed, unorthodoxy could have very serious consequences. Recent scientific debates have tended to be calmer, but being on the wrong side of an issue can still have career altering consequences (<http://www.mcfarlandbooks.com/book-2.php?id=978-0-7864-6301-5>). Computer science,

perhaps because it is young and well-funded, seems to have been relatively free of real schisms, but it still inspires energetic debates.

Computer science doesn't have a culture of retraction, and in any case many of these debates aren't about the kinds of mistakes that lead to retractions. The useful debates, the ones we can learn from, are those where both sides are publicly available in writing. These are a valuable instructional resource, and I sometimes assign them as reading to my students. They show an important part of science that often gets swept under the rug—students enjoy seeing that things aren't as cut-and-dried as teachers often try to make them sound. It's useful to try to figure out who's right and who's wrong. Alternatively, some debates are more about personalities, and still others feature researchers talking past each other based on their different assumptions and perspectives.

Considered harmful

An early debate, which feels a bit quaint now, began when Edsger Dijkstra published [Goto Statement Considered Harmful](#) (1968), which argued against unstructured control flow in programming languages. Follow-ups included [Goto Considered Harmful Considered Harmful](#) and [Goto Considered Harmful Considered Harmful Considered Harmful](#) (1987).

Multiple versions of the facts

One of my favorite public debates concerns *N-version* programming: a software-development method where several implementations of a specification are run in

parallel and voting is used to determine the correct result. If independent implementations have independent defects, this method would lead to a substantial increase in software reliability. John C. Knight and Nancy G. Leveson wrote a [paper](#) (1986) showing that the assumption of independent faults is suspect. This finding did not sit well with the proponents of *N-version* programming, and while I cannot find online copies of their rebuttals, Knight and Leveson's [reply](#) to the criticisms includes plenty of quotes. This is great reading, a classic of the genre.

Can we at least agree that CATOCS is a great acronym?

Starting in the late 1980s, Ken Birman's group was advocating causal and totally ordered [multicast](#): a primitive for building distributed systems that provides strong guarantees about internode communication in distributed systems. David Cheriton and Dale Skeen were less than impressed and wrote [15 pages](#) to that effect (1993). Birman wrote a [long response](#) to the criticisms. Also see [Neha Narula's later take](#) on the debate (2013).

File system performance evaluation is hard

A 1991 [paper](#) introduced log-based file systems, which increase the performance of writes to files by reducing the number of seeks needed to perform a file update. In 1993 Margo Seltzer et al. published a [paper](#) describing and evaluating an implementation of a log-based file system, with a [follow-up](#) in 1995. John Ousterhout, one of the authors of the original paper, [disagreed](#) with the evaluation. Seltzer and her coauthors [rebutted his critique](#), and Ousterhout had, as far as I know, the [last word](#).

You would not get a high grade for such a design

Another classic debate, [Torvalds vs Tanenbaum](#) [1992], was about how operating systems should be structured: as a monolithic collection of code running in kernel mode, or instead as a group of independent subsystems isolated by the memory management unit. Also see some (one-sided) [comments](#) on a reincarnation of the debate here. Related to this discussion, in 2005 Steven Hand et al. published [Are Virtual Machine Monitors Microkernels Done Right?](#) In response, Gernot Heiser et al. wrote a [paper](#) with the same title in 2006 but coming to the opposite conclusion.

A very obnoxious paper?

“Social Processes and Proofs of Theorems and Programs” is a provocative [opinion piece](#) written in 1979 by Richard De Millo et al. about the role of formal methods in software development. Dijkstra called it “a very obnoxious paper” (see [page 14](#) of a transcript of an interview with Dijkstra from 2001) and wrote a [response](#) called “A Political Pamphlet from the Middle Ages.” De Millo et al. replied: “We must begin by refusing to concede that our confidence in a piece of real software has ever been increased by a proof of its correctness.” See also the [letters to the editor](#) of *CACM* responding to this article, Victor Yodaiken’s take on the [debate](#), and [three more shots](#) fired in 2010 –two by Moshe Vardi and one by the original paper’s authors.

This guy’s arrogance takes your breath away

Dijkstra and John Backus had an (only partially public) spat in the late ’70s, written up [here](#).

SWATT or be SWATTed

The computer security research community has an especially strong tradition of refuting published results. For example, SWATT ([software based attestation](#)) offers a protocol for checking that a remote system has the memory image it is supposed to have. A 2009 [paper](#) called “On the Difficulty of Softwarebased Attestation of Embedded Devices” presents concrete attacks on SWATT. SWATT authors Adrian Perrig and Leendert van Doorn [did not agree](#) that the attacks were valid, and, finally, the paper’s authors, Aurelian Francillon et al., [responded](#) to the refutation.

A matter of integrity

CPI ([code-pointer integrity](#)) is a technique for avoiding control-flow hijacking caused by memory safety errors in C or C++ code. [Missing the Point\(er\)](#) [2015] presents attacks against CPI, while [Getting the Point\(er\)](#) [2015] argues in favor of the security of CPI.

ACKNOWLEDGMENTS

I’d like to thank many blog readers and Twitter users for providing feedback on the original blog post from which this article was derived.

John Regehr *is a computer science professor at the University of Utah, USA. He likes to create software tools for making software better.*

Copyright © 2017 held by owner/author. Publication rights licensed to ACM.