EDITORIAL

# Musings
## Theory of Mind

RIK FARROW

Rik is the editor of *;login:*.
rik@usenix.org

I am going to depart from the realm of computer science briefly, because I want to discuss a problem which is rampant in software design and papers. The problem involves the Theory of Mind (ToM), the "ability to attribute mental states—beliefs, intents, desires, pretending, knowledge—to oneself and others" [1]. But while ToM generally refers to interpersonal relations or philosophy [2], I am going to focus on the part about attributing knowledge to the people who will use your software or read your papers.

I was talking to an old friend, who had been complaining about his son. My friend said that, unlike his son, he just decided one day that he would focus on work and become responsible. If he could do it, so could his son.

I found myself suggesting that my friend look up Theory of Mind. Just because my friend could resolve to buckle down doesn't mean that other people would, or could, behave just like he did.

I remembered ToM from college psychology classes from many years ago. Today, deficiencies in ToM are now associated with autism among other disorders. That really isn't what I am referring to. Rather, thinking that because you did or know something, so should anyone else, just seemed a bit, well, not quite sane to me.

### Theory of Mind and CS

Where ToM and CS intersect is a bit different, having more to do with culture. As an editor, I am constantly running into this, as I read articles or papers where the authors assume that you have the same background and understand the same jargon that they do. After all, all the people they work with speak that jargon and have the same background information, right?

I've written an editorial about the importance of being able to write clearly [3], and ignoring ToM can mean rejected papers. You really shouldn't assume that people will just accept your brilliant research if you can't articulate it clearly.

I wrote a column about a software design issue many years ago [4] that dealt with ToM, but without saying so directly. I wrote that column because I had observed that most people had a difficult time with state machines. I found I could set friends' digital watches for them, even though they couldn't, because I understood the watches (and their two or three buttons) had different purposes depending on their current state. I've since discovered that clocks with four buttons, and no manuals, have so many states that even I have trouble setting them.

Today we get devices, such as smartphones, complete with state machines implementing the user interface. There are no manuals—what you need to do is find someone who has already communicated with someone who knows how the damn things work. Of course, the next update means that what you learned no longer works, and you need to make another social connection to understand the new interface. And speaking of overloaded interfaces, the most popular smartphone uses a single button that has a multitude of different purposes, depending on the software's current state. What an amazing design—for engineers.

Consider how this works in the place where the new interface gets developed. Someone comes up with some new UI widget and shows a coworker how to use it. The knowledge gets spread to others, and if the widget is compelling enough, it appears in the public version. But only insiders initially know how to make it work. It's like building systems where every new feature is an Easter egg [5].

## The Lineup

We have many articles related to cloud in this issue. We lead off with an article explaining the design goals and implementation of VFP, Microsoft's very different version of vswitch. I met Daniel Firestone during NSDI '17, where he presented the only industry paper, one which provides more implementation details about VFP.

Reid Priedhorsky and Tim Randles, of Los Alamos National Laboratories, describe their open source solution, Charliecloud. They determined that a lighter-weight solution than Docker would work best for HPC. To help maintain a familiar interface, containers are still built using Docker but are run via Charliecloud. Their article also helped me understand more about containers.

I interviewed James Bottomley. James has written about containers versus VMs for *;login:* [6], and I wanted to probe his viewpoints further now that he has changed jobs. James explains a lot more about the difference between containers and VMs, the Linux system calls used to set containers up, and why containers haven't been embraced by many vendors.

We have another article from NSDI '17. "Knockoff," by Dou, Chen, and Flinn, examines the tradeoff between recomputing data in the cloud and the cost of copying data. Hint: oftentimes, recomputing is both cheaper and faster.

In the system administration and SRE section, we have two articles. Carata, Chick, and Sohan describe Resourceful, a tool they developed for use in OS research at Cambridge and have now open sourced. You use the Resourceful API to instrument apps, allowing you to produce performance data from the kernel about very specific activities relating just to portions of an app.

Roy et al. explain how they instrumented servers and network hardware at Facebook and discovered how they could uncover subtle network problems faster than the current monitoring tools used. Their approach does rely on having a well-balanced workload to start with but should work in any well-tuned environment.

In the security section, Escobedo et al. talk about how the BeyondCorp team at Google worked to make the transition from traditional VPNs to BC easier for both current users and new hires. They share key insights and techniques into how others might smooth the migration of users to a very different method of application and server access.

Geoffroy Couprie and Pierre Chifflier reprise work they have done (and presented at the IEEE LangSec '17 workshop) about making existing software more secure. Rather than attempting the Sisyphean task of rewriting software from scratch, the authors focus on input parsers, using Rust, with a compiler that fails to compile dangerous code by accident, and `nom`, a tool that makes building safe parsers easier.

David Beazley tells us how surprised he was when he witnessed a Python programmer writing code concurrently with a testing framework. David explains how other Python programmers can take advantage of using a very simple technique to improve writing even very simple apps.

David Blank-Edelman has another edition of his "Flying Perl" series. Having read about a programmer who had used Python to answer the question "Which airports are closest to each other?" David shows us how to perform the same task in Perl.

Dave Josephsen wanted to show his coworkers the real value of being able to measure performance. Using Phaser.js for the visualization portion and Go for the server, Dave quickly threw together a tool (demonstrated with a YouTube video) that uncovered bottlenecks caused by unbalanced load.

Chris "Mac" McEniry has taken over the task of writing a Go column. Mac begins by adding TLS support to Kelsey Hightower's `gls`, something Kelsey had wanted to do when he wrote his column. But, as you will see, adding TLS, while easy in Go, deserves its own column.

Margo Seltzer has contributed to what we hope will be a new column on education. Margo has converted her operating system course at Harvard to use *flipping*. Flipping involves swapping the usual way that material is taught, so students begin with self-study, then work on classroom projects. Instead of lecture, which can leave many students lost, flipping means that students' questions and problems become the focus. By the way, I invite other teachers to contribute to future versions of the education column.

Dan Geer has written his annual column about the Index of Cyber Security. The ICS relies on polling security practitioners, and in turn these professionals get to see how the others in their field responded to questions about the same issues. Dan shares the answers about four different questions, including this issue's favorite topic, the cloud.

Robert Ferrell has hopped on the strike-back bandwagon. If your organization is under attack, why wait for government or professional assistance when you can launch attacks yourself against the presumed offender? Robert suggests a handful of attack tools that you can use.

# EDITORIAL

## Musings: Theory of Mind

Mark Lamourine has two book reviews this month. The first is about using CoreOS and the second about a short book on RESTful standards.

While it was my friend that got me going about the Theory of Mind, I do believe that it is relevant to most people. Theory of Mind applies when giving directions: for example, "Turn right where the Sinclair gas station used to be" relies on local knowledge about something that disappeared long ago. Where I live, you might still get instructions like "Turn left at the 'Y'," an intersection that is now a circle and hasn't been a 'Y' for over 25 years.

In the worlds of our own specialties, we also have the problem of insiders' knowledge. If we intend to communicate effectively, we can't assume that our audience knows what we do. If that were true, why would we even be addressing them? ToM, or rather, the assumption that others have the same knowledge or beliefs that we do, is an all-too-easy trap to fall into. Do us all a favor and write for your audience, not for your own in-group.


***References***

[1] Wikipedia, "Theory of Mind": https://en.wikipedia.org/wiki/Theory_of_mind.

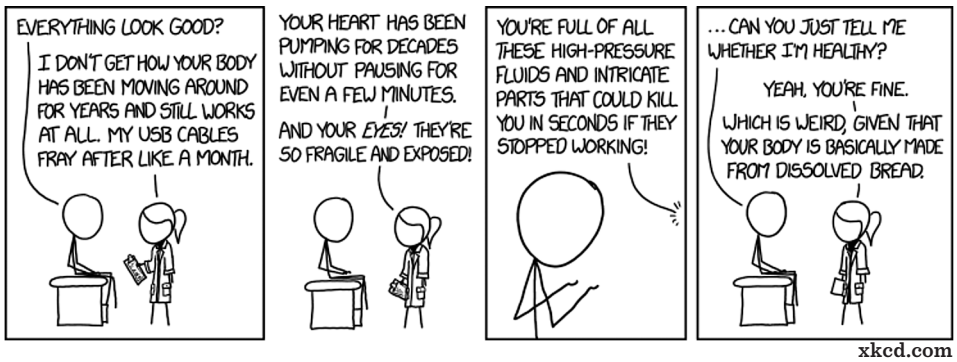
[2] Philosophy and ToM: http://www.iep.utm.edu/theomind/.

[3] R. Farrow, "Musings," *;login:*, vol. 41, no. 2 (Summer 2016): https://www.usenix.org/system/files/login/articles/login_summer16_01_farrow.pdf.

[4] R. Farrow, "Musings," *;login:*, vol. 23, no. 5 (August 1998): http://web.archive.org/web/20111110024139/http://www.usenix.org/publications/login/1998-8/musings.html.

[5] Definition of Easter egg: http://www.webopedia.com/TERM/E/easter_egg.html.

[6] J. Bottomley and P. Emelyanov, "Containers," *;login*, vol. 39, no. 5 (October 2014): https://www.usenix.org/system/files/login/articles/login_1410_02-bottomley.pdf.

# ENIGMA

# A USENIX CONFERENCE

## SECURITY AND PRIVACY IDEAS THAT MATTER

Enigma centers on a single track of engaging talks covering a wide range of topics in security and privacy. Our goal is to clearly explain emerging threats and defenses in the growing intersection of society and technology, and to foster an intelligent and informed conversation within the community and the world. We view diversity as a key enabler for this goal and actively work to ensure that the Enigma community encourages and welcomes participation from all employment sectors, racial and ethnic backgrounds, nationalities, and genders.

Enigma is committed to fostering an open, collaborative, and respectful environment. Enigma and USENIX are also dedicated to open science and open conversations, and will make all talk media freely available on the USENIX Web site.

**PROGRAM CO-CHAIRS**

Bryan Payne,
Netflix

Franziska Roesner,
University of Washington

**The full program and registration will be available in October.**

enigma.usenix.org

JAN 16–18, 2018

**SANTA CLARA, CA, USA**

usenix ASSOCIATION