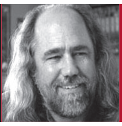
ON COMPUTING



Editor: **Grady Booch**
IBM, grady@computingthehumanexperience.com

All Watched Over by Machines of Loving Grace

Grady Booch

YOU ENTER A CAVE full of twisty little passages, all alike. You and your friends follow the path of a more experienced spelunker, who happens to be a bit portly. He gets stuck in a narrow corridor, unable to move forward or backward, thus trapping the rest of the party. Unfortunately for you, a stream on your side of the cave is rising. If you do nothing, you and your friends will drown. Fortunately for the guide, his head is on the dry side of the blockage, and he will continue to breathe. You have a stick of dynamite with you.

What should you do?

You could use the dynamite to kill your guide, thereby opening a way out for the rest of the party. This is a rather utilitarian philosophy, best summarized by Spock's catch phrase "The needs of the many outweigh the needs of the few … or the one." Killing your guide maximizes life, which we presume is a factor that's not unreasonable to optimize.

If your ethics instead embrace the point of view that all killing is wrong, then you might choose to do nothing, thereby letting the rising waters take their course. Your inaction might be in harmony with your ethics, but it would mean that you and your friends would die and only your guide would survive.

The Double Effect and Software

The ethical conundrum I've posed comes from a 1967 paper by Phillipa Foot.[1] Phillipa's thought experiment has been recast in modern times as the trolley problem.[2] and reformulated in a number of ways. No matter the variation, the center of this experiment attends to the doctrine of double effect, which explores the issue of whether it's permissible to intentionally carry out a harmful act to bring about a good result.[3] Depending on your moral center—and assuming you choose to act in integrity with that center—you would face the question of killing another human to save yourself.

At first glance, this might seem like nothing more than a topic for a late-night philosophical party conversation with friends, fueled by good food and strong drink.

But let's recast the question to make it more interesting.

What would a software-intensive system do? Or more precisely, what would you program a software-intensive system to do, or what would you teach it to do?

A semiautonomous drone will inevitably face this problem: should it terminate the terrorist it has targeted even if an innocent child suddenly enters the kill zone? A semiautonomous car will face a similar choice: if a pedestrian suddenly steps in front of the vehicle, should it swerve, knowing it will hit the car beside it, possibly seriously injuring multiple occupants?

Clearly, any discussion of a software-intensive system that actively takes a human life is an emotional subject, so let's dial back the scenario to something that's pure emotion, and reconsider the question.

Algorithms are essentially thoughtless.

Inadvertent Algorithmic Cruelty

Facebook. Ah, love it or hate it, it's undeniably the way that a billion or so people around the world connect. As an engineer, I respect the mantra that if it works, it's useful. On one hand, I love Facebook for the ability to stay in touch with true and intimate friends (the kind who will show up on your doorstep at 2 a.m. if you call them in need) as well as more casual ones (the kind for whom you will call the police if they show up on your doorstep at 2 a.m. but are still amused in following their media-documented journey through life). On the other hand, I detest it for the way its algorithms take away my control of what I want to see when I want to see it. I am at peace that I am part of Facebook's product content and so participate in Hobson's choice: I have the degrees of freedom to be a member of Facebook or not, and having made that choice in the affirmative, (begrudgingly) accept the consequences.

This is not to say that many of Facebook's features don't annoy me. From the perspective of best user experience practices, I'd judge the experience to be positively user hostile. Their "Year in Review" app is one of those things I find superficial and therefore ignore, but to some—such as Eric Meyer—its very presence is hurtful. As Eric explains in his blog, Facebook presented his Year in Review with a picture of his daughter—who had died earlier that year.[4] To Eric, this was a demonstration of "inadvertent algorithmic cruelty." Eric states the issue eloquently:

> *Algorithms are essentially thoughtless. They model certain decision flows, but once you run them, no more thought occurs. To call a person "thoughtless" is usually considered a slight, or an outright insult; and yet, we unleash so many literally thoughtless processes on our users, on our lives, on ourselves.*

He goes on to observe that "If I could fix one thing about our industry, just one thing, it would be that: to increase awareness of and consideration for the failure modes, the edge cases, the worst-case scenarios."

To say that algorithms are thoughtless is a reasonable and unemotional statement of fact. They have no moral center; they have no sense of right or wrong; they cannot take responsibility for their consequences. Bits cannot feel. However, we who craft such algorithms are expected to be thoughtful. Where does our responsibility begin, and where does it end?

More Dilemmas

The artist collective !Mediengruppe Bitnik (the ! is actually part of their name) developed the Random Darknet Shopper, a bot programmed to make random purchases of $100 in bitcoins every week in a darknet marketplace (https://wwwwwwwwwwwwwwwwwwwwwwwwwwww.bitnik.org/r). Over the past year, their bot has purchased cigarettes, counterfeit branded clothing, master keys, and drugs. Some of these purchases were legal, but many were not.

The good folks at the *Guardian* have asked the right question:

> *Can a robot, or a piece of software, be jailed if it commits a crime? Where does legal culpability lie if code is criminal by design or default? What if a robot buys drugs, weapons, or hacking equipment and has them sent to you, and police intercept the package?*[5]

In the case of the Random Darknet Shopper, Domagoj Smoljo, one of its creators, acknowledged that "We are the legal owner of the drugs—we are responsible for everything the bot does, as we executed the code. But our lawyer and the Swiss constitution says [sic] art in the public interest is allowed to be free."[5]

At least in this case there exists a legal safe harbor. In different legal jurisdictions, it might not be so. For example, Google is facing a case in Hong Kong over its ranking algorithms, which are programmed as well as learned.[6] Albert Yeung is suing Google because it offers up related search terms for his name that point to criminal gangs, a situation Albert considers hurtful to his reputation. As the High Court ruled, "Google 'recombines and aggregates' data through its algorithm and therefore can be legally regarded as a 'publisher,' meaning it may be sued for defamation."[6]

Let's consider one more software-intensive system. Event data recorders are the norm for all new automobiles. Privacy issues concerning use of that data abound—a topic for another column—but what about the case in which a driver willingly releases that data in real time to an insurance company to obtain optimal rates? My insurance company might bump my rates up if it finds me speeding, but what if I was speeding for an extended period because I had a sick child in my car whom I had to take to the hospital, or because I was trying to escape, as a victim of some road rage? The presumptive context of any algorithm would be the law; the real context that any human judge would see would entail compassion.

The dilemma here has been well covered—and certainly not well resolved—in the courts regarding mandatory minimums, wherein judges are given zero degrees of freedom to shape the punishment of a certain kind of crime to its particular context. Here we have already seen the societal implication of judgment without compassion; computing makes it profoundly easy to release algorithms without compassion.

Richard Brautigan, author of *Trout Fishing in America*, once wrote a poem, part of which reads

> *I like to think (it has to be!)*
> *of a cybernetic ecology*
> *where we are free of our labors*
> *and joined back to nature,*
> *returned to our mammal*
> *brothers and sisters,*
> *and all watched over*
> *by machines of loving grace.*[7]

From a theological perspective, judgment is often defined as receiving that which one deserves, mercy as not receiving that which one deserves, and grace as receiving that which one does not deserve. Richard published this poem in 1967, the year Foot proposed her ethical conundrum. In that age, we could only dream of such machines. Now, we build them. Furthermore, these practical ethical conundrums are only beginning to become fully manifest as we slowly and inevitably and irreversibly surrender to these machines of our own creation.

As software professionals, what should we do?

References

1. P. Foote, "The Problem of Abortion and the Doctrine of the Double Effect," *Oxford Rev.*, no. 5, 1967, pp. 5–15; www2.econ.iastate.edu/classes/econ362/hallam/readings/footdoubleeffect.pdf.
2. J. Clark, "How the Trolley Problem Works," HowStuffWorks.com, 3 Dec. 2007; http://people.howstuffworks.com/trolley-problem.htm.
3. "Doctrine of Double Effect," *Stanford Encyclopedia of Philosophy*, 2014; http://plato.stanford.edu/entries/double-effect.
4. E. Meyer, "Inadvertent Algorithmic Cruelty," blog, 24 Dec. 2014; http://meyerweb.com/eric/thoughts/2014/12/24/inadvertent-algorithmic-cruelty.
5. M. Power, "What Happens When a Software Bot Goes on a Darknet Shopping Spree?," *The Guardian*, 5 Dec. 2014; www.theguardian.com/technology/2014/dec/05/software-bot-darknet-shopping-spree-random-shopper.
6. S. Lau, "Hong Kong Tycoon Can Sue Google over 'Autocomplete' Search Suggestions, Court Rules," *South China Morning Post*, 7 Aug. 2014; www.scmp.com/news/hong-kong/article/1567521/hong-kong-court-rules-tycoon-can-sue-google-over-autocomplete-search.
7. R. Bratigan, *All Watched Over by Machines of Loving Grace*, Communication Co., 1967; www.brautigan.net/machines.html.

GRADY BOOCH is an IBM Fellow and one of UML's original authors. He's currently developing *Computing: The Human Experience*, a major transmedia project for public broadcast. Contact him at grady@computingthehumanexperience.com. Follow him on Twitter @grady_booch.

See www.computer.org/software-multimedia for multimedia content related to this article.