

CREATING HUMANE CODELAW

Gene Koo

“By using codelaw to carry out policy, government shoves analog pegs into round holes, resulting in the same loss of fidelity that occurs when music is ripped into digital formats.”

As a lawyer and techie at Massachusetts Law Reform Institute in the mid-2000s, I became aware of a computer system called Beacon used by the MA Department of Transitional Assistance (aka “welfare”) to distribute various benefits such as food stamps to Massachusetts residents. Occasionally, our clients would have their benefits reduced or cut off because of errors in Beacon programming, and our advocates would fight not only to restore their aid, but to fix the system.

What was happening in Massachusetts was happening around the nation, and indeed our errors were relatively benign by comparison. In Colorado, faulty software generated hundreds of thousands of incorrect benefits calculations, and in New York the state’s benefits distribution

system was so egregiously broken that our colleagues there brought suit in federal court and won sweeping changes.

These are some of the mundane but vitally important ways in which software is becoming the mechanism whereby government executes laws. It's not hard to find other examples, from "deadbeat dad" lists to terrorist "no fly lists" to the inner workings of voting machines, tax calculators, and even Predator drones. (Professor Danielle Citron, to whom I owe much of the following analysis, has documented many more examples.) So perhaps Lawrence Lessig's profound observation, *code is law*, has a corollary: *law is code*. That is to say, if software is increasingly the guise under which laws manifest in our daily lives, it behooves a democratic society to begin treating that software as law.

Software that executes law ("codelaw") presents a number of challenges to a democracy. The simplest are bugs, coding errors that lead to wrong results. Bugs present relatively easy cases: like potholes, if you find them, you fix them. As with potholes, the reality may be harder—a common excuse we heard was that the state just didn't have the money to hire someone to patch the software—but in principle everyone agreed that these problems should be fixed.

The larger democratic challenge arises when codelaw isn't so much *wrong* as it is *not necessarily right*: while it may not contradict the law, neither is this particular implementation the only way to construe the law. In short, the software assumes a particular *interpretation* of an ambiguous law, and in so doing, it essentially *makes* law.

By using codelaw to carry out policy, government shoves analog pegs into round holes, resulting in the same loss of fidelity that occurs when music is ripped into digital formats. The 20th-century administrative state in America relies on a particular cascade of power, carefully tweaked to ensure democratic accountability: the elected legislature passes law; an administrative agency, with public input, promulgates rules to implement that legislation; and agency workers carry out the

rules. The gradual replacement of agency workers with codelaw reveals the cracks in this system. Because legislatures lack time and expertise to tight-fit laws, they delegate specifics to agencies for further rulemaking. But rulemaking isn't comprehensive either: nuanced decision-making still resides in agency workers who interpret and apply the rules. Codelaw takes discretion out of the hands of human beings.

Eliminating discretion can be good governance: people are notoriously susceptible to bias, corruption, and just plain meanness. The real problem for democracy is the gap between the round curves of human laws and the sharp edges of computer code. Agencies have traditionally promulgated rules expecting people to fill the gaps later. With codelaw, the people who fill the gaps are not trained government employees, but software developers, often with no substantive knowledge of the law nor accountability to the general public.

So what can be done to ensure that increasingly automated codelaw remains accountable to the people?

First, software should be fully open for inspection. Democracy depends on laws and rules being accessible to the people; codelaw should be no exception. But because only the best-resourced lobbyists can bug-check machine code, mere transparency is not enough. There must be meaningful participation.

Existing principles that cover traditional (legal) code offer guidance on handling codelaw. For example, most state and federal rulemaking require a period of public "notice and comment," during which concerned citizens can offer input. A publicly accessible quality assurance cycle would create a parallel process for codelaw. So when Massachusetts prepares to release Beacon 2.0, it should enable people like my colleagues at MLRI to submit tricky food stamp scenarios to test that the software gets the right results.

In the long run, new forms of "semantic code" that's both human-readable and machine-executable may narrow the gap between fuzzy

legislation and binary software. “Legalese” as a software language may not deepen public confidence, but it would at least enable more precision for lawmakers.

Finally, we need a more nuanced understanding of the appropriate role of codelaw. Deployed properly, software can ameliorate systemic human failings such as sexism and susceptibility to scams. But conversely, we should also recognize the limits of software, and identify the aspects of governance best entrusted to thinking and feeling human beings. Codelaw may herald a terrifying dystopia where machines arbitrarily decide our fate. But it also invites us to imagine a world where software augments our greatest capacity for just, compassionate, and human governance.

About the Author

Gene Koo is a Fellow at the Berkman Center at Internet & Society, where he researches a variety of topics from open education to moral values embedded in video games. Prior to Berkman, he worked for Massachusetts Law Reform Institute, where he developed informational websites for lawyers and the general public, and also helped launch the Center for Legal Aid Education. He holds a J.D. and B.A. from Harvard.