Seven Ways in Which Code Equals Law (And One in Which It Does Not)

Cindy A. Cohn / James Grimmelmann

"Code equals law:" what might this statement mean? As our laws and our computers become increasingly intertwined, clear thinking about the relationship between code and law becomes increasingly important. In the early days of the Internet, many computer programmers were convinced that code would make law irrelevant. More recently we've seen legislators, regulators and judges who seem convinced that law can and should dictate code. Neither absolutist view is likely to prevail—virus writers go to jail and DeCSS is still available worldwide—but maybe by exploring the connections between the two concepts we can see how each can teach the other something. In addition, perhaps we can learn a few things about how both can help us create and preserve freedom as we move deeper into the digital era.

A first connection between the two words comes straight out of the dictionary. The Oxford English Dictionary says that a code is "[a] systematic collection or digest of the laws of a country." From the *Code Napoléon* to the California Motor Vehicle Code, lawyers are more than comfortable with the idea that the contents of a "code" constitute "law."

Taken literally then, the first way "code equals law" is a tautology, because "code" is defined to mean "law".

But people usually mean something more when they compare code and law, because they have in mind a different definition of "code." The OED hasn't entirely caught up with this definition, but Merriam-Webster says that "code" can mean "a set of instructions for a computer." It's not a coincidence, people say, that we use the same word for computer code as for legal code: these two kinds of code really are alike, deep down.

The idea that people have here is that programmers and lawmakers are doing the same thing, playing the same game: they're both *coders.* Programmers code computer systems; lawyers code legal systems. The difference between these systems is no more fundamental than the difference between the Java and Scheme programming languages, or between common-law architectures and civil-law ones. Coders are people who write in subtle, rule-oriented, specialized, and remarkably complicated dialects.

A second meaning for "code equals law," then, might be a kind of pun: computer code is nothing more or less than legal code transplanted to the electronic realm.

There is something to this view, something more important than just an esthetic observation about two crafts. It emphasizes that programmers and lawmakers are both *designers.* Good code, in all its forms, partakes of the virtues of good design, virtues such as clarity and cleanliness, utility and elegance, transparency and stability. The U.S. tax code is not well designed: its knotted mass of overlapping provisions and special-case shelters conceals a remarkable amount of unfairness. Show the code to a programmer and she will be able to diagnose it as a pile of one-off kludges slapped on top of kludges on top of kludges all the way down.

Design values should matter in law: many of the issues with which the EFF deals stem from instances of bad design in the law, places where coders of all stripes can see that something is wrong. The Digital Millennium Copyright Act has many faults, but among them is its *inelegance.* One of the reasons the DMCA has brought forth so many "gotcha"

lawsuits—it's been used to frighten professors and attack products from toner cartridges to garage-door openers—is that its provisions are ambiguous and confusing. It has been expanding into a general law to prevent tinkering and competition, even though it was passed (and passed off) as a more narrowly targeted bill to deal with large-scale copyright infringement. With clearer statutory text, Congress might not have created such opportunities for misuse, or at least would have been held more accountable for doing so. The DMCA is a case of the legislative coding process gone wrong.

So a third meaning of "code equals law" is as a reminder that law is a kind of code, that lawyers can learn from the insights of programmers into the design process.

This interpretation hinges on the pun between "legal code" and "computer code;" it works because the later sense of "code" grew out of the former. According to the OED, in the 19th century the word came to mean not just a specifically *legal* code, but any specialized system of rules, especially one that used specified words to stand in for other words. In the 20th century, this was the meaning the early computer pioneers had in mind when they picked "code" as a term to describe the artificial languages they were using to program their new inventions. The missing link between "computer code" and "legal code" is "secret code." The common theme here is *linguistic:* codes are written systems of communication.

This way of looking at code has been an important one for me and for the EFF: one of the most important freedoms we defend is the freedom of speech, and we're particularly emphatic that the protection given to speech shouldn't depend on the language or medium used. My first major case was helping a math professor, Dan Bernstein, challenge and overturn the encryption export regulations that prevented him publishing a computer program online. Professor Bernstein wrote a cryptographic computer program called "Snuffle" in the C programming language. In order to start the peer review process crucial to all science, he sought to publish Snuffle on a Usenet newsgroup called sci.crypt. The U.S. government told Professor Bernstein that if he published the code, he could go to jail as an arms dealer.

In order to clear the way for Professor Bernstein, we first had to convince the court that the publication of Snuffle was a speech act for purposes of First Amendment protection. We did so, and when the court then compared the export restrictions with the precedents and values protecting speech, the restrictions were struck down. From that first case, the legal holding that code is speech has gained acceptance by every other court that has considered the question, although we learned in the *2600* case that some courts will still find ways to rule against the publication of speech even after they acknowledge its existence. Nonetheless, I believe that this foundation linking code to speech will ultimately help ensure that the Internet remains a place of broad freedom of expression. I also believe that it is correct in two separate ways.

On the one hand, when computer scientists want to convey their ideas, the most natural and obvious language they have available is code. This is fundamentally what Professor Bernstein was doing. Telling a computer scientist or a mathematician that she can't use code in discussing her work would be like telling literary scholars they can't use French or economists that they cannot use graphs or formulas. The effect would be to restrict scholars to inferior forms of communication. Code is speech; restrictions on the distribution of code are restrictions on speech.

On the other hand, code is a speech-enabler, just as the printing press and pen and ink are. Code today provides some of the best and most important outlets for free speech. When we say that the Internet turns anyone into a publisher, we're really saying that *code* turns anyone into a publisher, whether it's Front Page or Blogger or Eudora. Ill-considered restrictions on the use of code can all too easily take away that remarkable free-

dom. This is why EFF has opposed technical mandates, whether they arise from faux standards bodies such as Broadcast Protection Discussion Group (BPDG), or legislation like that proposed by Senator Hollings last year, inartfully named CBDTPA. And just as code can enable speech, it can also squelch it if not carefully crafted. One other issue the EFF is starting to worry about is overly-zealous anti-spam solutions that aren't carefully tailored to protect wanted speech even as they attempt to identify and stop unwanted messages. Lately we've seen an increasing number of noncommercial e-mail lists struggle against anti-spam dragnets that seem not to recognize the importance of preventing this unintended consequence. Noncommercial listservs have been a godsend to small, underfunded speakers who are trying to reach a willing audience. Spam is clearly a problem, but if we want to keep free speech thriving online we need to recognize how, even unintentionally, code can hurt speech as much as it can enable it.

All of this is "code equals law" in a higher sense of "law." Free speech, both including code and as mediated by code, is a fundamental component of the kind of democratic society that stands behind law as we know it. And while many of the major legal decisions and regulations applying it have originated in the U.S., the values that undergird the conclusion that code is speech under U.S. law should support the same conclusion when the applicable standard is Article 19 of the International Covenant on Civil and Political Rights or the free expression provisions that exist in nearly every national legal structure.

This kind of free speech is only becoming more and more essential because we are all coders now. Some people code in C++; others code in HTML—but there are also those who "code" in a language of pull-down menus and icons. I'm not a programmer, but when I turn on my computer and start opening applications and doing things with them, I'm doing a type of coding, too. I'm teaching my computer to do the things I want it to do by giving it instructions in a specialized vocabulary. If there ever was a line between "coders" and "users" of technology, that line has long since disappeared.

It's wonderful that different people use computers in different ways. Some people like to write device drivers for their toasters; others just think it's useful that someone else wrote one. So it's absolutely vital that people have the *freedom* to use computers in all in the different ways they want to, subject only to the outermost limitations. Redrawing the limits of that freedom with a line between code and speech that no longer exists in real life means criminalizing some of the most innovative and creative things people who love technology, really love it, are doing. Tell most people that they can only be users of code and we'll never see the next generation of coders, because we'll have taken away the freedom that teaches people to love technology and its possibilities.

The Free Software movement understands this point: think of the slogan "free as in free speech," which links the free exchange of ideas to the process of software development. What a great idea: that software can be both *used* and *rewritten* by anyone. Code is as much a medium of expression for programmers as clay is for sculptors, and digital painters and electronic musicians are coders, too. Some of the most profound recent advances in parallel computing and computer graphics have come from moviemakers intent on creating the most compelling special effects possible. Activists are making political points with their choice of domain names and in their use of JavaScript. Code feeds speech and speech feeds codes.

Thus, a fifth sense in which "code equals law" would be that, increasingly, code is the *subject* of law. Code is speech is one of the first of those. But with code penetrating so much of society, more and more laws and legal decisions will become decisions about code.

But here is where law and computer code diverge. Law is a process run by people as

well as rules. Laws don't put criminals in prison or award damages by themselves: it takes police officers and prosecutors, judges and juries, to make a legal system work. All of these people have their own vision of their proper role in the system; all of them have some degree of control in shaping its outcomes.

A legal code is just another input to the process—a particularly important input, to be sure—but in the end, just one more factor all of these *people* draw upon when deciding whom to compensate and whom to punish. The terminology reflects the truth: to "codify" law means to write it down in one place, not to make law in the first instance.

A sixth interpretation of the claim that "code equals law" would say that the statement is false, because while a code is a kind of law, "law" embraces much more than just "code." Law is a basic tool of social order; it is a reflection of a society's values and also the most important way a society puts those values into practice. Law shapes the institutions and the environment within which people live their lives; it embodies millions of decisions about what it means to live and work in society. It does all of this in one of the most direct ways imaginable: by telling people what to do and punishing them unless they follow instructions.

Misguided legal procedures can send innocent people to jail and leave torture victims without remedies. The Patriot Act makes librarians into secret government agents and removes key limits on electronic surveillance. A judge who finds copyright infringement can order all copies of a book seized and pulped. These cases aren't troublesome just because the laws could have been better; they're troublesome because the laws could have been better and people suffered as a result. In the end, only law professors really care whether laws are elegant. Practicing lawyers and ordinary citizens have much more basic concerns: are the laws fair? Are they just? Do they work? Do they cause collateral harm?

Law regulates conduct. Criminal laws against arson, fraud, and murder keep people from doing deliberate harm to others; contract law holds people to their promises. Tort judgments deter people from taking undue risks at the expense of others; water law keeps people from taking more than their fare share out of rivers. Law shapes what people do and don't do, what they are encouraged to do and what they are afraid to do.

Law isn't the only thing that regulates. Other institutions—from markets to gossip, from speed bumps to professional associations—also regulate conduct. They prescribe rules of behavior; they have ways of enforcing their prescriptions. One of the most important such institutions, one becoming more important every day, is of course code. Code determines what you can and can't say in an email message; what your computer will and won't do; who can see your web page, and how. Code allows some things and forbids others. It makes possible all of the new virtual spaces and new forms of interaction we associate with the Internet and other technologies, but it also lays down rules for those spaces, puts limits on what forms those interactions are allowed to take.

This is the seventh, and most important, way in which "code equals law:" in spaces controlled by code, the code *fills the role* usually associated with law. At EFF, one of our founding principles was "architecture is policy," and it's still our core view.

Some people, seeing this connection, and remembering the values of good code, try to improve the legal system by treating it as a computer. People come to me with ideas for hacking the law. "The government says that cryptography is a weapon," they say, "but the Bill of Rights says we have the right to bear arms. So that means we have a Constitutional right to use cryptography."

But the legal system isn't a computer. If you can't convince a judge that what you're proposing is consistent with the values underlying a law, your argument will go nowhere. People go to jail every year because they think they've found a way to hack the Sixteenth Amendment. "The income tax is illegal," they say, or, "The income tax is voluntary, see, it says

so right here," and then they get convicted of tax evasion and sent to jail. We did convince several judges about the Constitutional dimension of cryptography, but the claim started from the values of the First Amendment, not a mechanical reading of its words.

It's a category mistake to treat the legal system as just another architecture with its own specialized language. Code and law are *different* ways of regulating; they have different textures. All of those people who are required to make the legal system work leave their mark on its outcomes: they make a certain amount of drift and discretion almost inevitable. Code doesn't have such a limit: it can make perfectly hard-nosed bright-line rules and hold everyone in the world to them. Code is capable of a kind of regulatory clarity and intensity that law can only state, never really achieve.

But it's a *good* thing that law can't do these things and usually doesn't try. Successful legal systems have values, important values, that don't always translate into the rarified abstractions of computer code. I like the judicial branch; I like explaining technical issues to judges and seeing them *understand* what's at stake. Once they see that, they will work with you to find ways to create a fair result. Good laws work with this flexibility: they explain what's at stake, and what needs to happen, and then they leave behind a little space, so that the legal system—all those prosecutors and judges and juries, all those *people*—can make the law work well in practice.

One of the reasons that American free speech jurisprudence has been so successful and such a popular symbol for Americans over the years is that it has this right kind of flexibility built in at its most basic level. The basic command of the First Amendment—"Congress shall *make no law* … abridging the freedom of speech"—is both absolute and vague about the details. The doctrine has evolved to deal with the demands of different ages: think about what a disaster would have resulted if the Constitution tied free speech to the technology available two hundred years ago. We can convince judges that code is speech precisely because the First Amendment left the definition of "speech" somewhat vague. At the same time, the absoluteness of that "no law" has made clear that once we know what "speech" means for a given age, we have to be scrupulously even-handed about protecting it.

Contrast the sorry state of our copyright law. Copyright's assumptions are still grounded in the technological assumptions of an earlier age—chief among them that copying works was expensive and time consuming—and those assumptions are reflected in the law in quite specific detail. The bulk of the text of the Copyright Act in the United States is a long series of very detailed regulations about copies of print, audio and audiovisual recordings and their specific distribution schemes. It's not that these regulations were necessarily bad ones at the time: you could make out a case that their precision and exactitude were virtues. But now they're inappropriate; they're getting in the way of the real goals of copyright: compensating artists and sharing creativity with the public. Law that looked less like code would have served us better.

The most dangerous laws are the ones passed by lawmakers who've forgotten that the legal system doesn't work according to the rigid and reliable logic of a computer. These are the laws that seem to be straining against the vitality of language itself, filled with detailed and counter-intuitive definitions. And these are also the laws that seem to think that every aspect of society can and should be programmable: that with the right code everything will be perfect.

This vision of law and society as computers, awaiting only the right instructions, is what drives policymakers to blindly bulldoze vibrant neighborhoods to put up bleak concrete towers under a theory that this is more efficient housing, to chop down forests and replace them with trees planted in straight rows for easier watering. And it's what drives politicians to insist that technological innovation conform itself to the wishes of copyright hold-

ers or other particular interests. The broadcast flag proposal currently before the Federal Communications Commission would require digital television receiving devices to respond to a bit in the broadcast signal indicating how the broadcast may be used. This isn't bad when thought of as computer code: it requires only that computers and consumer electronics devices be coded in certain ways. But it would be truly terrible as law, because the requirement that devices contain certain codes (and do not contain others) is antithetical to the basic "freedom of code" that has led to our growing use and development of technology.

Law, done right, has certain values that are worth preserving. Thousands of years of experience have taught lawyers a few lessons worth remembering. No law with which a majority of the people consistently disagree can be maintained for long. There must always be space for *de minimis* exceptions to a rule of prohibition. Important corrections and exceptions will become apparent only in hindsight, as problematic cases come up. Changes in technology and society will require subtle changes in the meaning and application of rules. There is no one right way to solve any problem.

To the extent that code is acting as law in spaces and ways that are increasingly important to our increasingly technological societies, code will need to respect, as best it can, these deep legal values. Sometimes this will mean *not* doing everything in code: one of the reasons DRM and the broadcast flag proposal are so worrisome is that they threaten to eliminate the flexibility we associate with fair use law and technological progress. At other times, this will mean wiring these values into the code itself. The end-to-end design that has made the Internet so successful is, in some sense, just the legal value of humility, as written into code. We don't know what people will do, so we won't try to solve their problems for them before we even know what those problems are.

So this is an eighth and final meaning of "code equals law:" a forceful reminder that as code takes on the powers of law, it must also take on the responsibilities of law. Lawyers today are learning about code and they can benefit much from this knowledge, but coders will also need to become lawyers, in the best sense of the term, if we are to preserve freedom online and build the kind of digital world where we all want to live.