12 ethical dilemmas gnawing at developers today

As software takes over more of our lives, the ethical ramifications of decisions made by programmers only become greater

By Peter Wayner
Contributing Editor, InfoWorld | APR 21, 2014

The tech world has always been long on power and short on thinking about the ramifications of this power. If it can be built, there will always be someone who will build it without contemplating a safer, saner way of doing so, let alone whether the technology should even be built in the first place. The software gets written. Who cares where and how it's used? That's a task for somebody in some corner office.

More troubling: While ethics courses have become a staple of physical-world engineering degrees, they remain a begrudging anomaly in computer science pedagogy. Yet as software takes over more of our life, the ethical ramifications of decisions made by programmers only become greater. Now that our code is in refrigerators, thermostats, smoke alarms, and more, the wrong moves, a lack of foresight, or downright dubious decision-making can haunt humanity everywhere it goes.

What follows are a few of the ethical quandaries confronting developers every day -- whether they know it or not. There are no easy answers, in some measure because the very nature of the work is so abstract. To make matters worse, business has become so inextricably linked with computer technology that it is difficult to balance the needs and motivations of all invested parties in trying to prevent today's business-case feature from becoming tomorrow's Orwellian nightmare.

The trick is to think past the current zeitgeist and anticipate every future utilization of what you build. Pretty simple, huh? Consider this less of a guidebook for making your decisions and more of a starting point for the kind of ethical contemplation we should be doing as a daily part of our jobs.

Ethical dilemma No. 1: Log files -- what to save and how to handle them

Programmers are like pack rats. They keep records of everything, often because it's the only way to debug a system. But log files also track everything users do, and in the wrong hands, they can expose facts users want kept secret.

Many businesses are built on actively protecting log files. Some remote-backup services even promise to keep additional copies in disparate geographic locations. Not every business aspires to such diligence. Snapchat, for example, built its brand on doing a very bad job of backing up data, but its users are attracted by the freedom of the forgetful system.

The mere existence of log files begs several ethical questions. Are they adequately protected? Who has access? When we say we destroy the files, are they truly destroyed?

The crux is ascertaining what information is worth keeping, given the risks of doing so, ethical or otherwise. Here, the future complicates the equation. In the 1960s, smoking was widely embraced. No one would have thought twice about keeping track of people's smoking habits. Today, however, the knowledge of someone's smoking activity can be used to raise health insurance rates or even deny coverage.

Future business deals; future government regulations; an unforeseen, desperate need for new revenue streams -- it may be impossible to predict what seemingly innocent log file will become problematic in the future, but it is essential to consider the ethics of how you handle the logs along the way.

Ethical dilemma No. 2: Whether -- and how -- to transform users into products

It's a well-worn adage of the startup era: If you're not paying for a service, you're not a customer; you're the product.

On the Internet, "free" services abound. In fact, the question of where the money will come from is often put off, being off putting. We just build the amazingness, keep an eye on the adoption metrics, and figure someone else will take care of the dirty work of keeping the server lights on. Worst case, there are always ads.

Developers need to be upfront about who will support their work and where the money will come from. Any changes should be communicated to users in a clear, timely fashion to avoid shock and blowback. Transforming people into products is an ethical shift not to be taken lightly. Shady ad deals, shady ad networks -- we need to be careful how we handle the implicit trust of early adopters.

Ethical dilemma No. 3: How free does content really want to be?

A number of businesses depend on serving up content without paying those who create it. Some turn around and sell ads or even charge for access. These businesses often couldn't survive and couldn't price their material as attractively if they had to shoulder their fair share of the development costs. They develop elaborate rationalizations about "sharing" or "fair use" to cover up an ethically shaky decision.

Developers must ask themselves how their code will support everyone in the food chain, from creators to consumers. Do the people creating the content want their work to be distributed this way? Are they happy to work for exposure or attention alone? Are they given a fair share of the revenue?

Not considering these questions amounts to turning a blind eye to piracy. After all, not all information just "wants to be free."

Ethical dilemma No. 4: How much protection is enough

Some say that everything should be double-encrypted with two different algorithms and locked in a hard disk that is kept in a safe. Alas, the overhead slows the system to a crawl and makes development 10 times more onerous. To make matters worse, if one bit gets flipped or one part of the algorithm is wrong, the data is all lost because the encryption can't be undone.

Others don't want to lift a finger to protect the data. The next team can add special encryption later if it's necessary, the developers might say. Or they might argue that there's nothing sensitive about it. Teams that ignore these responsibilities are usually able to generate plenty of other code and create piles of wonderful features that people crave. Who cares if they're secure?

There's no simple answer to how much protection to apply. There are only guesses. More is always better -- until the data is lost or the product doesn't ship.

Ethical dilemma No. 5: To bug-fix or not to bug-fix?

It's hard enough to negotiate the ethical shoals when they involve active decisions, but it's even harder when the problem can be pushed aside and labeled a bug that will be fixed eventually. How hard should we work to fix the problems that somehow slipped into running code? Do we drop everything? How do we decide whether a bug is serious enough to be fixed?

Isaac Asimov confronted this issue long ago when he wrote his laws of robotics and inserted one that forbid a robot from doing nothing if a human would be harmed through the robot's inaction. Of course his robots had positronic brains that could see all the facets of a problem instantly and solve them. The questions for developers are so complicated that many bugs go ignored and unfixed because no one wants to even think about them.

Can a company prioritize the list fairly? Are some customers more important than others? Can a programmer play favorites by choosing one bug over another? This is even more difficult to contemplate when you realize that it's hard to anticipate how much harm will come from any given bug.

Ethical dilemma No. 6: How much to code -- or compromise -- to prevent misuse

The original Apple Web camera came with a clever mechanical extra, a physical shutter that blocked the lens when it was off. The shutter and the switch were linked together; there was no way to use the camera without opening the shutter yourself.

Some of the newer webcams come with an LED that's supposed to be illuminated when the camera is activated. It usually works, but anyone who has programmed a computer knows there may be a place in the code where the camera and the LED can be decoupled. If that can be found, the camera can be turned into a spying device.

The challenge for the engineer is anticipating misuse and designing to prevent it. The Apple shutter is one of the obvious and effective examples of how it can be done elegantly. When I was working on a book about cheating on the SAT, I met one hacker who was adding networking software to his calculator. After some deliberation, he decided to only support wired protocols because he was afraid kids would sneak a calculator with Wi-Fi into an exam. By supporting only wired protocols, he ensured that anyone in a test would need to run a wire to their neighbor's machine. He hated skipping the wireless protocols, but he felt the risk of abuse was too high.

Ethical dilemma No. 7: How far to defend customers against data requests

If you collect data, it's a safe bet that your organization will someday be caught between serving your customers and serving the government. Requests to deliver data to legal entities are becoming increasingly common, leaving more and more software and services organizations to contemplate to what extent they will betray their customers' privacy before the law. You can scrutinize these requests and even hire your own lawyers to contest whether they are truly lawful, but the reality is that the courts will be debating legalities long after your funding runs out.

There are no easy solutions. Some companies are choosing to leave the business rather than lie to their customers. Others are trying to be more open about requests, which the government often tries to forbid.

Ethical dilemma No. 8: How to deal with the international nature of the Internet

The Internet runs everywhere, avoiding many of the traditional barriers at the borders. This can be a recipe for legal headaches when customers A and B are in different countries. That's only the beginning, because servers C and D are often in entirely different countries as well.

This leads to obvious ethical issues. Europe, for instance, has strict laws about retaining personal information and views privacy breaches as ethical failures. Other countries insist on companies keeping copious records on dealings. Whose laws should a company follow when customers are in different countries? When data is in different counties? When data is transferred across international lines?

Keeping up with every legal contingency can be Herculean, leaving many organizations surely tempted to bury their heads in the sand.

Ethical dilemma No. 9: How much to give back to open source

Everyone knows that open source is free. You don't pay anything and that's what makes it so wonderful and complex. But not everyone contemplates the ethical issues that come with using that free code. All of the open source packages come with licenses and you need to follow them.

Some of the licenses don't require much sacrifice. Licenses like the Apache License or the MIT License require acknowledgement and that's about it. But other licenses, such as the GNU General Public License, ask you to share all your enhancements.

Parsing open sources licenses can present ethical challenges. One manager from a big public company told me, "We don't distribute MySQL, so we don't owe anyone anything." He was keying on the clause, written decades ago, that tied the license's obligations to the act of redistributing software. The company used MySQL for its Web apps, so he felt it could take without giving back.

There are no simple ways to measure the ethical obligations, and many programmers have wasted many keystrokes arguing about what they mean. Still, the entire endeavor will grind to a halt if people stop giving. The good news is that it's often in everyone's best interest to contribute because everyone wants the software to remain compatible with their use of it.

Ethical dilemma No. 10: How much monitoring is really warranted

Maybe your boss wants to make sure the customers aren't ripping off the company. Maybe you want to make sure you get paid for your work. Maybe some spooky guy from the government says you must install a backdoor to catch bad guys. In every case, the argument is filled with assurances that the backdoor will only be used, like Superman's powers, to support truth and justice. It won't be used against political enemies or the less fortunate. It won't be sold to despotic regimes.

But what if the bad guys discover the hidden door and figure out how to use it themselves? What if your backdoor is used to support untruths and injustices? Your code can't make ethical decisions on its own. That's your job.

Ethical dilemma No. 11: How bulletproof should code really be

But what if the bad guys discover the hidden door and figure out how to use it themselves? What if your backdoor is used to support untruths and injustices? Your code can't make ethical decisions on its own. That's your job.

Ethical dilemma No. 11: How bulletproof should code really be

Sure, the minimal calculation, simple data structure, and brute-force approach works well in demo when the problems are small. The users try out the code and say, "Gosh this works quickly." Several months later, when enough data has been loaded into the system, the cheap algorithm's weaknesses appear and the code slows to a crawl.

Developers must often decide exactly how hard to work on the final product. Do you whip off a quick and cheap solution or spend weeks adding bulletproof code that deals with extreme cases? True, clients and users should assume some of the responsibility during the requirements and sign-off phases, but developers are often better equipped to anticipate potential contextual hiccups of running code.

Ethical dilemma No. 12: How much should future consequences influence present decisions

Many projects don't make waves. The information goes in and never escapes. Some, however, take on a life of their own, escaping into the wild where they may do untold harm. Security, penetration testing, espionage -- these are obvious candidates for considering the collateral damage of your code.

Take Stuxnet, a virus widely considered a tool for attacking the centrifuges used to purify uranium in Iran. Perhaps it succeeded, but now it lives on, floating along in Windows systems throughout the world.

For most developers, the collateral damage is less obvious. We code for today -- a hard enough proposition -- but we should also consider the future.

Some programmers, for example, love to write complex code that integrates with the operating system and installs new or more complicated drivers. Is that going to work in the future? Will it play well with other new drivers? Will it work with the next generation of the OS? Or will your software leave people with a computer that runs slower and fails more frequently even when your software isn't running?

It may seem simple, but choosing whether to stick with the APIs or follow the standards is an ethical decision. Yes, technology is evolving quickly, and a slavish devotion to outdated mandates can be a hindrance to progress. But we need to consider our part in writing code that lasts a bit longer and not take the decision to swim outside our lane lightly. We can always ask for changes to the standards or the APIs if we need them.

https://www.infoworld.com/article/2607452/12-ethical-dilemmas-gnawing-at-developers-today.html