Ethics

Software engineering as a profession has a shockingly immature grasp on the ethics of engineering, and the impact we're making on the world. In other engineering disciplines, ethics and codes of conduct are baked into the professional bodies that represent practitioners. Software engineering as a discipline has largely sidelined professional bodies, preferring best practices and coordination to emerge naturally, rather than from a professional body. It's as rare to come across a chartered engineer in a digital organization as it is to come across one without (or not working towards) the qualification in civil engineering.

The pros and cons of these professional bodies can be debated at length, although the consensus seems to be that they do not add much value, and qualifications (certainly for developers) do not hold much weight with employers. In some countries, "engineer" is a protected term, and only qualified individuals can use it—similar to medical doctors, or lawyers. This is to protect the image of an engineer, doctor, or lawyer as someone who can be trusted. In these countries, software engineers are often referred to as software developers instead.

A defining feature of a profession is that a professional should be someone you can trust, and this is either achieved by baking it into the training of that professional (the most famous being the Hippocratic oath for medical professionals) or by either legislative or industry-led registration and certifications (in the UK, for example, a CORGI registered gas fitter implies a high degree of trust, which is especially important when dealing with something as dangerous as gas).

The one thing these approaches to ethics have in common is that people who undertake roles in these professions aim to do no harm. The software development industry has a much less formal route to entrance than many other professions, and no widely recognized certification schemes. Combined with its lack of professional bodies (which have their own ethical codes), this is a potentially dangerous situation, as we are not effectively self-regulating either.

There have been some visible efforts to address these shortcomings, the most visible of which are the Geek Feminism and diversity in technology movements, but these are only focused on a few aspects of ethical behavior. The Code Craftsmanship movement similarly focuses on a set of ethics around the approaches taken to actual coding, but doesn't address the whole breadth.

Ethics is an important component of any society, and are generally most effective when they emerge as practices through consensus rather than through legislation. So, what does an ethical code look like for a software developer? At its core, it is the same as the ethics we should be practicing as individuals in society, and the more formal codes of other professions: we should strive to do no harm, or at least minimize harm.[1]

Although on the surface it may seem easy to achieve this—after all, if a shopping app crashes due to a sloppy engineering approach, it won't have the same impact as a bridge collapsing (though of course, there are many safety-critical software systems too)—but the impact our apps have on society is often more abstract. This does not make it less important to consider.

Privacy

One of the major tensions personal computing has introduced has to do with privacy. Computers, and especially mobile phones, have become intensely personal devices, and they and the services they interact with hold a wealth of data relating to users, who likely won't want to share it with everyone. As a professional, you are expected to respect your users' privacy, but at the same time, this data can be valuable to an organization and inform your decisions to build a better product or increase effectiveness of strategies such as advertising.

An analogy that often arises in relation to user data is that of oil, often focusing on the perceived value of data and how it's desirable to "mine" it. When refined, oil is very useful and valuable, but the flip side is that in its crude form, it is toxic, and if spilled, can destroy entire ecosystems. Collecting and processing data should be considered risky, and the easiest way to reduce that risk is to reduce the amount of data you are collecting. This gets especially tricky when some level of personal data is technically necessary—for example, IP addresses could be considered personal information, but an IP address is required for users to establish a connection to a server.

Just because you *can* collect data does not always mean that you *should*. A "big data" craze has perpetuated the idea of collecting all the data you possibly can and then seeing if any trends emerge from it. However, the process for analyzing all of that data is complex, and for many organizations, this dream has never fully been realized. Instead, collecting more focused and meaningful data is both easier to analyze and easier to practice ethically.

Legislators are also increasingly reacting to what is perceived as a failure of the industry to self-regulate, and are implementing new laws to protect citizens from over-collection and questionable use of their data. The EU's General Data Protection Regulations (GDPR) has called into question the business practices of many online advertising providers and organizations that rely on advertising for revenue, and are forcing large-scale changes to those practices. Still, many consumers are surprised at how widely their data is being sold and shared, and the lack of oversight of the practice. This was further underlined by the scandal of Cambridge Analytica, which surreptitiously collected Facebook profile data of Facebook users, which was then used to target ads as part of political campaigns, including the Trump presidency and the Brexit vote.

When determining the necessity of collecting data, you must also think like an attacker. What would happen if the data you're collecting was leaked, or your company got acquired by another one? The reason you're using the data now may be ethically acceptable, but in another context, it might be morally dubious. For this reason, it's important not only to get consent from a user to collect their personal data, but consent also to process it in certain ways, and to record this consent explicitly alongside the data. Ensuring that this consent is tightly scoped can protect you from changes in your organization’s mission or policies, although an attacker will not respect this consent for processing. You may decide that, in those cases, holding data at all is too risky. For example, charities dealing with refugees may decide to limit the data they collect in case a hostile government administration subpoenas it to deport those refugees.

Even when anonymizing data, care must be taken, as it is possible, given enough data points, to still link some data to a specific person. One method for anonymizing data is to give users a random ID that is not linked to their actual identity, and then tie personal data to that. This method is more accurately described as pseudonymity, as people still have an identity within your system—it is just now tied to the random ID rather than one that relates to their individual identity. Given enough data points, it can be possible to tie this back to an individual. One famous example of this was in 2006,

when AOL released the search logs of users of their search engine, tied to anonymous IDs. However, looking at the search terms was sometimes enough to tie those anonymous IDs to an individual, because users will sometimes search for their own name, or for businesses geographically close to them, schools their children attend, etc. A number of users were able to be positively identified by journalists for press purposes.

This dilemma can be difficult for large organizations, especially where people may interact with them in many different ways, as this results in more data across the whole organization which, if correlated, could identify a user. A developer on a product team may be hard-pressed to confidently say whether or not their users' data is being used responsibly across the entire organization, but should be able to ask the important questions.

True anonymity would mean that no data points in your system are tied to any sort of identifier. This allows you to still develop aggregates and averages, but not correlations. For example, if you were examining the cities where people had items shipped, then you might be able to tell that people from Slough made more orders from your website than people from Ashford, but you would not be able to tell that the people from Ashford actually spent more in total, as you couldn't correlate across different data points.

Expectations of privacy change across time and cultures. The early Internet had a much lower expectation of privacy, and the people using it left much more information open to the outside world than they do today. In some cultures, sharing a constant feed of photographs of yourself traveling is acceptable and encouraged, whereas other cultures would shocked by that behavior, as it would reveal that your house is empty, putting you at risk of being robbed. Fully understanding all of the cultures you operate in and their expectations, as well as legal ramifications, is a necessity in the modern world.

For many, privacy is expected at all times, but for others, there are cases where a lack of privacy is acceptable. The cultural relationship between the state and its citizens is one example of this, highlighted by stereotypical differences between political parties and between places like the US and Europe. On one hand, many perceive the role of the state as that of a helping hand that cares for the citizenship, and having good data allows it to do that better. Others believe that the state should be minimized, and individual freedom trumps all, so the state collecting data on a large scale is an invasion of privacy.

An example of this is the UK's National Health Service (NHS), where a program known as "care.data," which aimed to link together a citizen's interactions and health records, was rolled out. Privacy activists campaigned against this, and many individuals were opposed to it, as they saw privacy as a core principle to be upheld. The NHS

believed that this system would not only improve individual care (for example, patient records would be available at any hospital), but allow data to be shared with researchers to monitor health trends of a whole nation, and possibly detect new correlations between different lifestyles and health conditions. The cost-benefit decision here was difficult, but ultimately privacy won over and the NHS shut down the scheme, although many health researchers regret the result.

This tension, like many others in ethics, does not have a straightforward answer, but requires careful reflection as you grow in your career. It is worth taking some time to reflect on the NHS care.data example given above to determine which argument you find more convincing and how you might respond if it were your own personal data at stake, perhaps on a smaller scale inside your own organization.

Cognitive Load

"Your app makes me fat" was the claim made in a famous blog post by Kathy Sierra,[2] who highlighted an academic study showing that people who have to do more difficult tasks (hence, have a higher cognitive load) show less willpower (for example, the ability to reject unhealthy food) than those who do simpler tasks.

This simple observation is very important when it comes to our role in developing the solutions that people use. We should aim to decrease the cognitive load for our users to minimize any negative consequences of using our applications. Good usability also aims to achieve this, but sometimes can conflict with business goals. Sometimes, you may be asked to design a user interface that puts the needs of the business above the needs of the user. These are sometimes called dark patterns, and include things like making it hard to delete an account, or hiding an option to decline travel insurance when purchasing a flight. As professionals, we have a responsibility to push back on features that have negative consequences for the people who use what we build.

Similarly, we also have to think about the human cost of our products. If we're building an app for employees, then they may potentially be spending a considerable portion of their day working with your app, and you can have a large impact on their life. This goes beyond technical decisions to include entire product decisions, especially in the case of "gig economy" apps, where we have a professional responsibility to treat the

---

users of the apps (for apps like Uber, this is the rider as well as the driver) with respect, and to avoid misleading them or making their life more difficult. Ultimately, these users are still individuals, and this category of product can have a significant impact on their lives.

There are entire classes of products that developers may have to make a personal ethical judgement on. For example, the potential for working on military systems may

require careful consideration about whether or not to accept a job, but projects that may seem more benign, like a social network, could be used to cause harm if used incorrectly by its owners. Ethical consequences can be subtle too. For example, a face recognition app may help a social network user quickly and accurately tag photos, but if used by a totalitarian state, it could be used to persecute citizens.

There is no right answer to these questions, but they are issues that as professionals we must consider, and re-consider if the context of our work changes. We not only have the right, but also the responsibility, to say "no" to our work being used in unethical ways.

Energy Usage

The design of Bitcoin's blockchain is mathematically brilliant, and has certainly garnered a lot of attention, but it is flawed in an engineering sense. As the blockchain is decentralized, a transaction has to propagate through the blockchain for it to appear in the record of truth. At the time of writing, each transaction uses the same kWh of electricity as a typical house uses in a day.

The expenditure of energy, and other physical resources, is certainly a visible part of the impact IT has on the world, but it also feels very abstract from the perspective of an individual developer, especially with the move to the cloud. When organizations ran their own data centers, energy usage was at least visible to that organization, often as a cost to be minimized, in addition to any environmental impact. In the cloud, it's hard to see this impact, much less consider it, but it is important to do so.

Of course, there are good engineering and business reasons to minimize energy usage as well; increasing performance/efficiency and reducing costs can be big drivers too. Evaluating whether or not unnecessary load and resources are being used, and whether or not capacity is sufficient (is much of your server capacity being left idle for a long time?) can go a long way.

Trust

The people we work with, whether that be clients, colleagues, or users, place a high degree of trust in us to do our jobs. They trust that we will do what we say we will, usually for an agreed price, and for us to do that to the best of our abilities.

They also trust that we do not misrepresent ourselves, and behave honestly. Selling yourself as an AWS expert after launching a single EC2 instance on the free tier, or as a Ruby specialist with several years of experience when in reality all you've done is follow the Rails tutorial, undermines that trust. We should not be scared to admit it if we truly believe that we don't meet the expectations of those who ask us to do something, although, at the same time, stepping out of our comfort zone to try something new is okay as long as those expectations are made clear.

We should also trust our colleagues, and be trusted by them. Undermining, playing politics, or bullying is completely unacceptable— yet all too common—in any profession. When a team undertakes a retrospective, the prime directive is a good principle to follow in most interactions with people: "we assume that everyone did the best job they could with the knowledge they had at the time."

Diversity is a factor here, too. Technology is not quite reminiscent of scenes from *The Wolf of Wall Street,* but there are still too many pockets where "culture fit" (i.e., looking like the people who are already on the team) is valued above skill set. This still happens explicitly, but unconscious bias is even more dangerous. Research has shown that more diverse teams perform better than monocultures. Many organizations are capitalizing on this by mixing functions on a digital team, but it is also our responsibility to make technology a welcoming industry for all who want to contribute, and to foster teams where all members can contribute to the best of their ability.

Software engineers also have a fairly unique style of interaction with our peers through the mechanism of open source. Open source is often the result of volunteered time, and the result of this work is often made available under fairly permissive licenses. Some of these, like the famous MIT or BSD licenses, are completely permissive, whereas others, like the GPL, are more nuanced, and require that any changes or work that builds on top of their work is also made available in a "share-alike" way if they are distributed.

What the GPL enforces in its license often becomes a societal expectation of people who use more permissive licenses. For example, taking an open source component and selling it for profit is a fairly rare occurrence, as to do so would jeopardize the entire nature of open source. Using an open source component as a library within a

commercial product is often seen as acceptable, but any improvements or changes made to the library itself are expected to be contributed back to the main project, so others can benefit from them. Web development blurs the lines, as code is not shipped directly to users as a product, but the same responsibility is still there: if you make improvements or fix bugs in an open-source library, the societal expectation is that you will contribute those back. Together as an industry, we can then make each other better, and avoid betraying the trust placed in us by those who make these libraries available, who then might turn to alternative methods such as digital rights management, or just not make their work available at all.