Introduction to Software Engineering (ISAD1000/5004)

9. Ethics and Professionalism

Updated: 26/05/2020.

So far we have discussed how software engineering happens, at a technical level.

This topic is different. Ethics is personal, cultural, ideological. It concerns how human beings[a], not just software projects and software systems, can come to harm. This can, of course, be an uncomfortable subject.

We have to discuss it now, because when else would we? Learning a skill, or an entire profession, puts you in a position of power. Your future colleagues, and the users of your software, will be relying on your work. People you'll never meet will be affected by it. Every good act you perform will be amplified by this power, but every mistake can be as well, if you're not careful.

Everyone makes mistakes, and everyone deals with the consequences. But certain kinds of mistakes are special. Certain kinds of mistakes can harm people in ways that cannot be fixed. 99% of us innately want to avoid this. Even the remaining 1%[b] might be persuaded that doing right by others is ultimately in their own self-interest. So it's important to understand how and why.

1 Moral Philosophy

Moral philosophy is vast and enduring, and can also be a deeply interesting area of inquiry. I am not remotely a moral philosopher[c], so I cannot give you an in-depth look at this field.

It is useful to be aware of the basics, though. Moral philosophy attempts to define what morality/ethics[d] is, and how to tell whether something is ethical or not. It seeks to understand the entirety of the concept of morality/ethics. However, in doing so, there turns out to be a lot of debate and disagreement. Who knew human beings could *disagree*?

1.1 Deontology

Deontology translates to "duty" or "obligation". Here, we try to define morality/ethics as the adherence to a set of rules. So, to be ethical is to follow the rules [5, pp. 284–286].

What *are* the rules? Well, that's the million-dollar question. Moral philosophy doesn't just hand out pre-written sets of rules. You still have to figure that part out yourself, but you can imagine the kinds of rules you're likely to have:

- Don't inflict violence on others;

[a] Not *just* human beings, but anything we may value for its own stake: animals, the environment at large, and human culture, for instance.

[b] I'm not suggesting that the split is actually demonstrably 99:1. These numbers are just for illustrative purposes.

[c] The "Ph" in "PhD" stands for philosophy, but for most PhDs this is a vestigial trait, like the human appendix or the tiny leg bones of snakes. Science originated in philosophy, but today's philosophers deal in questions that cannot be answered with empirical evidence or mathematical modelling, such as morality.

[d] What's the difference between "ethics" and "morality"? Some people swear there is one, but it's very difficult to express in a simple, definitive way, and is not important for our purposes here.

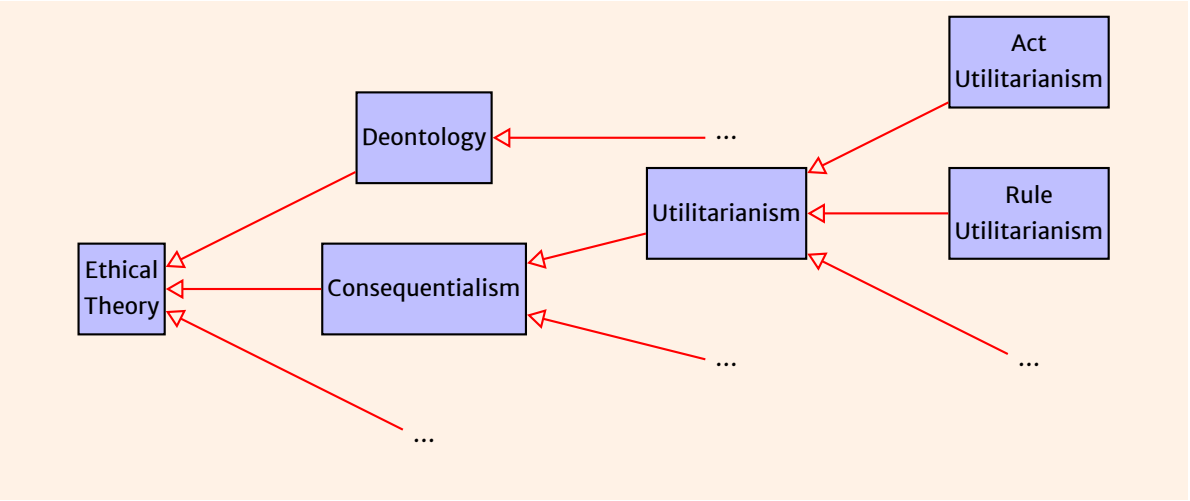

Figure 1:
We don't all agree on what ethics actually is.

- Don't break your promises;
- Don't steal;
- Don't play loud music at 2am;
- etc.

The rules may ultimately be a personal choice, although they're likely to overlap with laws, organisational policies, cultural norms and religious edicts. Lots of people have written lots of rules over the millennia already.

Moral philosophy does try to put some order to them. You cannot just have any rules. You must attempt to have rules that treat everyone as equal beings, and which can be generalised to all situations. For obvious reasons, you should also avoid having contradictory rules.

However, this approach comes up short for two reasons:

- You can't come up with rules to cover every possible situation.

  You may agree *after the fact* that it was immoral to dump a large quantity of truth serum into the water supply, but you probably didn't have a rule for that beforehand[e].

- The rules that you do come up with are not always going to seem sensible all the time.

  In reality we *do* allow certain kinds of violence, in cases of self-defence, or defence of others, or detention of suspects by the police, or euthanasia, or surgery, or contact sports, etc. We might also forgive stealing (at least from a moral point of view) if the thief was able to save their own life, or the lives of others, by doing so. Suddenly our rules seem a little inadequate.

  (You may be aware of the hippocratic oath

1.2 Act Utilitarianism

Utilitarianism (part of the slightly broader concept of **consequentialism**) judges ethics/morality based on the **utility** of what actually happens as a result. Specifically, **act utilitarianism** considers the consequences of each individual action that you may (or may not) perform [5, p. 287].

There are no rules. There is simply utility, which is a measure of "how good" the outcome will be.

An action can, in fact, inflict some amount of harm, provided it does more good than harm. We seek the "greater good", or the net utility once all the positive and negative consequences have been summed together. This only applies to the consequences of one action though. You cannot build up a reserve of "ethicalness" that then lets you do harm while remaining ethical.

[e] If you did, I salute you, but there will certainly be *some* things you haven't thought of.

Utilitarianism addresses the problems mentioned with deontology. Provided you define utility in a broad-enough way, utilitarianism does deal with situations and exceptions that you haven't anticipated.

However, act utilitarianism also has problems. To be ethical, you must understand the consequences of your actions with some accuracy. This seems a reasonable requirement in most circumstances, but not all. What about young children, or adults with dementia? Or what if something sudden and *unpredictable* happens? Say you give way to another person walking in the street, and they consequently become distracted by your arm gestures and get hit by a bus. You should have known. In the crudest utilitarian terms, you've just committed murder.

This problem also happens in reverse. Say you plant a bomb in a crowded place, but it fails to detonate because, at the last minute, a bird lands on it and chews through the wiring. You're off the hook. No harm was done.

After these sorts of examples, it may occur to you that rule-based ethics isn't a *completely* bad idea.

1.3 Rule Utilitarianism

Rule utilitarianism looks a bit like a combination of the previous ethical theories. There are rules to follow, but the rules must have positive utility. The rules are created so that, if consistently obeyed, they do more good than harm [5, p. 288].

This can help address the problems of the other ethical theories, but it also suffers from both their weaknesses as well.

You need not evaluate the exact consequences of absolutely everything you do. You're allowed to take mental shortcuts, by performing actions that *on average* result in more good than harm. You're not allowed to plant a bomb, because on average there are bad consequences. You are allowed to give way to people, because on average that's fine (and is just as likely to save someone from being hit by a bus as to be the cause).

But think about this. There's a lot of room between (1) trying to predict every consequence of every action you take, and (2) not thinking about them at all, except when deciding on the rules in advance. One might be unachievable in practice, but the other seems negligent. What if you have an unexpected situation where, nonetheless, you can see approximately what's going to happen?

1.4 The Trolley Problem

The **trolley problem** makes the murky nature of ethics even murkier. It is a lose-lose thought experiment that illustrates how difficult ethical dilemmas can be.

The "trolley" is an out-of-control tram (running on train tracks), and there are five people tied to the tracks, who cannot escape, and who will die if the trolley hits them. You can only do one thing to stop this: you can pull a lever, which will divert the trolley onto a different track, but there's also one person tied to *that* track, who will therefore die if you pull the lever.[12]

Do you leave five people to die, or do you save them by killing one other person? Utilitarianism says that you should save the five at the expense of the one. One death is less bad than five.

But say that you stumbled onto this situation as it unfolded. If you hadn't been there, the five

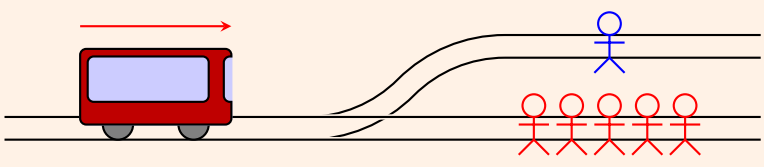

Figure 2:
The trolley problem.

would simply perish, and this would not have been your fault. But, by choosing to pull the lever, you are intentionally killing someone who was otherwise not really involved[f]. Whatever the situation, deontology would say that you have no right to take a life.

And the trolley problem is just getting started. We can imagine all sorts of horrible variations:

- What if the five people are near-death anyway, due to medical conditions, and the one person is a healthy child? Or, conversely, what if there were five children and the one person was near-death? Does a person's probable remaining lifespan affect the value of their life?
- What if the five people are criminals, and the one person is innocent? Should your own ethical determinations take into account other people's ethics, or lack thereof?
- What if the five people are strangers, and the one person is a close personal friend of yours?
- If the examples so far are steering you towards the "just don't kill people" view (i.e., deontology), then what if the one person caused the five people to be tied up in the first place, and may do so again in the future?
- And, even if you decide that all lives have equal worth, what if you weren't absolutely sure that the five people would actually die? Say that the track is going uphill, and hence the trolley is slowing, but you can't tell whether it's enough to avoid a collision. Meanwhile, the alternate track turns and goes downhill, so the one person would definitely die if you pull the lever.

  Would you kill one person on a gamble that five people *might* otherwise die?

The point of all this is to demonstrate the depth of the ethical dilemmas you can face. It is extremely difficult to have a consistent ethical theory that gives you seemingly-sensible guidance all the time.

But you also can't get out of the game. If you try to ignore the trolley problem, because it feels like a no-win situation, that implies you wouldn't deal with it in real-world either. If some version of the trolley problem really happened, and you just walked away, then that's a choice. You can't not make a choice, and you can't escape responsibility for it.

Imagine a version of the trolley problem with a hundred different tracks, where you can divert the trolley onto any one of them. Some of those tracks have no people tied to them at all[g]. Ignoring the problem may be the worst action you can take. If you're there, making a decision, you could save lives.

Thus, the only way to do good (relatively speaking) is to at least *try*, to put effort into thinking about what's right and why. You may not get it right all the time, but you'll hardly ever get it right if you don't confront the problem.

2 Mistakes and Harm in Software Processes

While moral philosophy attempts to discover the nature of ethics/morality, its high-minded debate isn't exactly a plan for *being ethical*. The understanding and scholarship of ethics, and the practice of ethics, are not exactly the same thing.

To be ethical, to do good instead of harm, you must understand how your actions could lead to

[f] We could also set up the situation so that the one person *isn't* tied down at all, but is simply walking along the track listening to music, doesn't believe the track is in use, and just won't see or hear the trolley coming.

[g] Call me a relentless optimist for envisaging a train track without any people tied to it.

one or the other[h]. You need foresight, imagination. You need to visualise how your effects on the world might ripple outwards from you, over time, and impact other people.

This can be specialist (discipline-specific) knowledge. The harms that might result from software engineering are different from those resulting from other fields, like medicine, law, politics, etc., although there are some overlaps.

We'll approach this by example[i].

Scenario A: Planning

You and a couple of colleagues are playing planning poker to determine how long you expect various project tasks to take. However, the client is pressuring you to get the project finished very quickly, and your colleagues are annoyed at you in general for previously producing estimates they thought (without good reason) were too high.

Therefore, since you're stressed, you pick lower estimates than you would normally, and these become part of the project schedule. Your company signs a contract with the client to deliver the software within a short timeframe.

During development, it becomes obvious that you underestimated the time required. The developers (not those who were producing the estimates) must now put in significant overtime for several months to try to get the job done. Some have to miss important family events, and others develop clear signs of work-related anxiety and depression. In the end, the project is not completed on time anyway, and the client refuses to pay.

Scenario B: Functional Requirements

You are doing the requirements analysis for a government website to help people apply for disaster relief payments. Each applicant enters their name, contact details and evidence of financial/property damage, and a government agency will then make payments to them accordingly.

In creating a use case for this, you write extensions that reject invalid names; i.e.:

> **Extension 2A:** the applicant's name contains an offensive word or non-letter characters.
>
> 1. The system asks the applicant to correct their name.
> 2. The use case resumes at step 1.
>
> **Extension 5A:** the applicant's name does not match the records obtained from the tax department.
>
> 1. The system asks the applicant to correct their name.
> 2. The use case resumes at step 1.

The developers write the website code according to these specifications.

As a result, your website makes it impossible for people who *really do have these names* to submit an application. They cannot "correct" their name, as it must match the name they've used for tax purposes, which your system considers invalid.

---

[h] We are assuming that consequentialism/utilitarianism will feature somewhere in our mix of ethical beliefs. It's not just going to be deontology all the way.

[i] I'm writing these scenarios in the "second-person" point of view, saying "you" as though *you* are the culprit. I don't mean to put any guilt on your shoulders, but I do intend to convey a sense of personal responsibility. Each of us personally carries ethical responsibility, not just some arbitrarily-named other person.

So, for certain victims of natural disasters, there is no way for them to receive assistance. They become homeless and destitute.

Scenario C: Non-Functional Requirements

You are developing a system to be used by police departments to keep track of ongoing criminal investigations. The system will be used remotely by multiple officers, each of whom will need to log in using a username and password.

You consider writing a non-functional requirement to have two-factor authentication ("2FA") as part of the login process, to help ensure greater security. This would mean that officers would also need to enter an additional randomly-generated number. However, you believe that doing so would cause too much inconvenience.

Shortly after the system is deployed, a novice hacker breaks in by "credential stuffing". The hacker purchased a list of passwords on the black market, previously stolen from other websites, and one officer was reusing the same password on multiple systems.

The hacker finds personal details of witnesses in an organised crime case, and posts these on the Internet. One of the witnesses is then tracked down and murdered by a contract killer.

Scenario D: Project Management

You are developing a new social media platform (similar to Twitter/Facebook/etc.), and using Scrum to do it. After each sprint, the latest version of the product goes live, so that users can immediately access the new features. Through good fortune, your application becomes surprisingly popular very quickly.

However, there is limited room for new features in each sprint backlog. In the beginning, you focus on what you believe to be the core functionality: basic messaging, friend recommendations, advertising, notifications, and video. Other features, including reporting or blocking of abusive content, are in the product backlog but are considered lower priority. They won't make it into a sprint backlog for several months.

The problem becomes clear when one user makes a controversial post that attracts the attention of many thousands of other people, including those on other social media platforms who then decide to make accounts on your system. This user begins receiving dozens, then hundreds of abusive messages every day, including graphic rape threats and death threats.

The current state of the system leaves no means to counter this, except to log off or wait months for the promised reporting/blocking features. But the user cannot do this without also losing access to their circle of friends and work contacts. Their daily life becomes psychological torture.

Scenario E: Testing

You are developing an ambulance dispatch system, to receive ambulance requests and notify drivers when and where to go. You don't write comprehensive unit tests, because the application seems to work fine anyway, and you can't see anything wrong with the code.

Unfortunately, there is a missing "`index += 1`" line in one of the `if` statements in the code. This isn't picked up in your informal testing, because that part of the code concerns what happens *after hours*, and you're only informally testing the system during work hours.

The missing line causes some records to be overwritten in an array/list (`records[index] = newRecord`). Consequently, a small number of calls to the ambulance service go missing, and an ambulance is never dispatched in those cases.

One person suffers permanent brain damage after a stroke, due to the resulting delay in getting to hospital.

These scenarios are hypothetical (though some are more hypothetical than others). However, by laying out the logical flow of events from mistake to harmful consequence, you can hopefully believe that these scenarios could happen.

The best defence against ethical mistakes is forethought and imagination. If you can contemplate the worst-case scenarios, and their possible causes, you can avoid them. You need more than a stern voice in your head saying "don't do that". You need a voice in your head that speculates about what might actually happen.

By definition, forethought comes before you do anything wrong. You can't wait to practice ethics. You don't get to experiment to see what happens. An ethical mistake is final. It cannot be fixed.

3 Professionalism

Professionalism is essentially about doing your job "properly". The assumption is that doing our jobs properly – acting as professionals – translates into being ethical, at least as far as the workplace is concerned. (This assumption may be challenged, as we'll see in Section 4.)

The scenarios in Section 2 are all arguably lapses in professionalism. They are cases where we didn't do our job properly, in one way or another. Here are some other things to watch out for:

Avoiding conflicts of interest. A conflict of interest happens when you have multiple goals or duties, and failing at one (particularly an important one) could help you succeed at another (often more personal).

For instance, if an athlete places a bet on their own competitor, they would have an incentive to lose the competition in order to win the bet.

Here are some examples in a software engineering context:

- You are writing software for the government to help track down tax evaders, but you're aware that your own family is guilty of tax evasion. (It's not just the tax evasion itself that is a problem. A conflict of interest is created by taking up the job, because you will have a strong incentive to do it badly.)
- Your company develops an operating system, and provides an online store where other developers can sell apps to run on your OS. The trouble is, you also want to expand your OS to include many of the features provided by these apps. If you do so, you would compromise the businesses of the app developers, who may have helped your OS become popular in the first place.

Challenging abuse to others. Unfortunately, you will have to deal with people in your life who are unethical and even abusive. Obviously *you* shouldn't be, but you also shouldn't just stand by when it affects other people.

I'm not going to lie about this: it takes guts to put yourself into the line of fire, to stand up for others. The person at fault may be someone you would normally respect, or someone with more power than you.

In some cases, standing up may be as simple as voicing support for the victim. But it may also require a "quiet word" with the abuser. In tougher situations, you may need to file a

formal complaint with your organisation. I'm not going to tell you that this will work, or that you'll be safe doing it. But, if you have that opportunity, it is right.

Challenging unethical instructions. Consider your response if you were asked, by your boss or someone in a position of power, to do something unethical. Similar difficulties apply to this as above.

Various organisations, such as the IEEE-CS[j] and the ACS[k], publish "codes of ethics", "codes of conduct", etc., to remind their members of their ethical and professional expectations. In certain situations, you may be legally required to uphold them, although of course ethics and law are not the same thing. Ideally, you should uphold them *irrespective* of whether you're likely to get into trouble for not doing so.

Note 1: The IEEE Computer Society's Code of Ethics (short version).

The following is the "short version" of the IEEE Computer Society Code of Ethics.[6] There is also a "full version" that expands on each of these points.

> Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:
>
> 1. PUBLIC – Software engineers shall act consistently with the public interest.
> 2. CLIENT AND EMPLOYER – Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
> 3. PRODUCT – Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
> 4. JUDGMENT – Software engineers shall maintain integrity and independence in their professional judgment.
> 5. MANAGEMENT – Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
> 6. PROFESSION – Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
> 7. COLLEAGUES – Software engineers shall be fair to and supportive of their colleagues.
> 8. SELF – Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

[j] The Institute for Electrical and Electronic Engineers (IEEE) Computer Society.

[k] The Australian Computer Society

Note 2:
The Australian Computer Society's Code of Professional Conduct.

The following is an extract of the ACS Code of Professional Conduct,[2] summarising its points. The full document expands on each of these points.

> As an ACS member you must uphold and advance the honour, dignity and effectiveness of being a professional. This entails, in addition to being a good citizen and acting within the law, your conformance to the following ACS values.
>
> 1. *The Primacy of the Public Interest*
>
>   You will place the interests of the public above those of personal, business or sectional interests.
>
> 2. *The Enhancement of Quality of Life*
>
>   You will strive to enhance the quality of life of those affected by your work.
>
> 3. *Honesty*
>
>   You will be honest in your representation of skills, knowledge, services and products.
>
> 4. *Competence*
>
>   You will work competently and diligently for your stakeholders.
>
> 5. *Professional Development*
>
>   You will enhance your own professional development, and that of your staff.
>
> 6. *Professionalism*
>
>   You will enhance the integrity of the ACS and the respect of its members for each other.
>
> In a situation of conflict between the values, The Primacy of the Public Interest takes precedence over the other values.

4 Are You Involved in the Right Thing?

While mistakes in SE processes, and lack of professionalism, can lead to awful outcomes, the *product* sometimes poses a bigger ethical dilemma. That is, even if everything goes right with the process, is the goal itself ethical?

Consider the following kinds of systems. I can't definitively tell you whether these examples are ethical or not. That's up to you. But (I think) it's fair to say that they're not ethical in quite the same way that, say, cooking lunch or playing the violin are ethical. You need to at least think about them.

- Software-controlled weapon systems, such as missiles, tanks, military aircraft, etc.[4] The risks here speak for themselves. These systems are explicitly designed to kill people, or act in support of lethal force. Even the *threat* of their use can be ruinous to vulnerable groups and nations. Software is typically used to make these systems more "intelligent", and hence more dangerous to those against whom they are used.

  Many would argue that, in the right hands, such systems can be used for legitimate defensive purposes and even potentially to liberate oppressed people. But you can see this is a delicate and high-stakes calculation.

- Surveillance software, including (but not limited to) facial recognition. Just as for weapon systems, your ethical calculations here probably depend on whether you think this software

will be used by the "right" people or not. Surveillance systems can reduce crime,[9] and facial recognition in particular may improve the efficiency of certain government processes.[7]

But surveillance systems may also create a problem for marginalised minority groups,[11] for whom the police and other authorities can act as adversaries as much as protectors. The argument that "you have nothing to fear if you have nothing to hide" misses a lot about human nature: both the simple human need for privacy, and the reality that power is routinely abused.

- Online gambling software. In essence, the gambling industry designs "games" that are not necessarily in the best interests of their own players. The goal of gambling is to win money by luck, of course, but the games are designed to thwart this in two ways. First, despite a small chance of temporary success, the player is essentially guaranteed to lose on average, over time. Second, the games trigger addictive behaviour, not just by accident, but by design.[10]

  Gambling may be defended. It is an individual choice, and the excitement of participation may be its own reward. Nobody is coerced into it. On the other hand, perhaps gambling services are manipulating and exploiting human psychological weaknesses (a kind of social engineering). In particular, for people who become addicted, limitless online gambling can ruin them both financially and socially.[8]

5 Do Good

With so many possibilities for doing harm, why do anything at all?

Software has enormous positive impacts on the world, and, with the right skills, you can continue this enterprise. You cannot do good without being involved.

In all of the scenarios in Section 2, you could just as easily be the one who *prevents* the harm being done, who imagines future impacts. And just as you might question the morality of certain kinds of software, there are many more examples of how it can improve the world:

- Weather forecasting is based on software, and while the information is probabilistic (and not *completely* accurate), it is accurate enough to significantly assist agriculture and other activities.
- Advanced driver-assistance systems can make driving significantly safer.[3]
- The entire internet is software-based, of course, and has (imperfectly) made most human knowledge accessible to most people, as well as providing cheaper, faster, more private (with some caveats), and more reliable communications.
- Digital video, audio and imagery has likewise (through the internet and consumer electronics) become accessible to everyone.
- Electronic payment systems make paying for goods and services trivial, even across continents, and also help reduce certain types of crime.[1]

This list is of course grossly incomplete. And you may yet think of software-based solutions to other problems that nobody else has thought of.

References

[1] Laura E. Armey, Jonathan Lipow, and Natalie J. Webb. The impact of electronic financial payments on crime. *Information Economics and Policy*, 29:46–57, 2014. **[See section 5.]**

[2] Australian Computer Society. Code of Professional Conduct. https://www.acs.org.au/content/dam/acs/rules-and-regulations/Code-of-Professional-

Conduct_v2.1.pdf, 2014. Accessed: 2020-05-25. [**See section 2.**]

[3] Simon Bickerstaffe. Seeing is believing. *Automotive Engineer*, pages 22–23, Sep 2012. [**See section 5.**]

[4] Kadir Alpaslan Demir. Challenges of weapon systems software development. *Journal of Naval Science and Engineering*, 5(3):104–116, 2009. [**See section 4.**]

[5] John F. Dooley. *Software Development, Design and Coding*. Apress, second edition edition, 2017. [**See sections 1.1, 1.2, and 1.3.**]

[6] IEEE-CS/ACM Joint Task Force on Software Engineering Ethics and Professional Practices. Code of Ethics. https://www.computer.org/education/code-of-ethics, 1999. Accessed: 2020-05-24. [**See section 1.**]

[7] Michael Koziol. 'world first': Government moves to radically overhaul Australia's international airports. *The Sydney Morning Herald*, 20 Jan 2017. Accessed: 2020-05-25. [**See section 4.**]

[8] Jan McMillen and Peter Grabosky. Internet gambling. *Trends & Issues in Crime and Criminal Justice*, (88), 1998. [**See section 4.**]

[9] Eric L. Piza, Brandon C. Welsh, David P. Farrington, and Amanda L. Thomas. CCTV surveillance for crime prevention. *Criminology & Public Policy*, 18:135–159, 2019. [**See section 4.**]

[10] Mike Robinson. Designed to deceive: How gambling distorts reality and hooks your brain. https://theconversation.com/designed-to-deceive-how-gambling-distorts-reality-and-hooks-your-brain-91052, Aug 2018. Accessed: 2020-05-25. [**See section 4.**]

[11] Katy Sian. Countering racism in counter-terrorism and surveillance discourse. *Palgrave Communications; London*, 3(1), Dec 2017. [**See section 4.**]

[12] Judith Jarvis Thomson. The trolley problem. *Yale Law Journal*, 94(6):1395–1415, 1985. [**See section 1.4.**]