Ethics as a skill of a software engineer?

H.-M. Järvinen
Professor
Tampere University of Technology
Tampere, Finland
E-mail: hannu-matti.jarvinen@tut.fi

ABSTRACT

One problem of the increasing intelligence of the systems is that the number of decisions having an ethical component is increasing, too. Traditionally, the designers of the system seldom made ethical decisions; the ethics was left for the user. However, when the system itself makes decisions, the ethical consequences have to be solved when the system is made. Since these decisions are mostly implemented by software, it is often the programmers that have to make these decisions.

Another problem is, if a programmer is asked to make illegal or unethical software. The well-known example of emission faking by car manufactures is an example of such software, but this seems to be far more common than this.

We have several ethic codes as the ethical guidelines for engineering. They give full ethical responsibility to engineers on the design and implementation of systems. The specific problem with software is that the programmers do not necessarily realise the ethical nature of the system; either they do not understand the application area enough to see it, or they do not know the full context of the piece of software they are implementing.

The conclusion discusses how to embed the awareness of ethical questions in software engineering education.

Conference Key Areas: ethics in engineering education, engineering skills

Keywords: software engineering education, ethics

1 INTRODUCTION

This paper is motivated from two phenomena that have recently become more evident. The first one is the increasing intelligence of the systems, and the second is that programmers have been told to make if not illegal, at least unethical systems.

The problem of the increasing intelligence of the systems is that the system makes bigger and bigger decisions autonomously. This has the consequence that the number of decisions having some ethical component is increasing, too. In traditional technology, the designers or manufacturers of the system seldom have to make ethical decisions; the ethics of product's usage is left to the user. The situation changes when the system itself makes a decision, and the ethical consequences have to be solved when the system is made. Since these decisions are mostly implemented by software, it is often the programmers that have to make them. From the viewpoint of this paper, one of the main problems is if the ethical aspect has not been noticed during the specification phase of the system, and the decision is left on the programmer, who might face difficulties in taking the responsibility of the decision that could be hard to answer even by experts on the application area. There is also the additional problem that even the programmer implementing the software might be unaware that there are some ethical questions involved.

The other problem of unethical software is that the programmers being asked to make illegal or unethical code more and more often. This has been discussed in network publications like Business Insider Nordic [1] and Freecodecamp [2].

The overall finding is that it is very difficult to see the possible ethical consequences of a technical decision. Hence, the studies of software engineering should embed ethics to allow students better recognise possible ethical questions.

The paper is organised as follows: First, some ethical codes are introduced in Section2. In Section 3, the problem of unethical code is discussed. Section 4 discusses the problems raised by the increased intelligence. The paper ends in conclusions.

2 ETHICAL CODES

There is a number of ethical codes intended to be used in engineering or software development. Archimedean Oath [3] originates from Lausanne (1990), and it is intended as a general guideline for all engineers. ACM and IEEE have provided ethical codes for their members, but also these are general in nature [4], [5]. These codes emphasise personal responsibility of the engineer. In Archimedean Oath, this is expressed as follows:

"I shall recognize the responsibility for my actions, after informing myself to the best of my abilities, and shall in no case discharge my responsibilities on another person."

In the case of software, it is sometimes hard to recognise if the code has any ethical consequences. The person writing the code may not know the circumstances it will be executed, and even the description of the required code might be written in a way that this is impossible to find out. Furthermore, more and more software is built on existing frameworks, components, libraries etc., and the developers of the framework can hardly be responsible for applications that use the framework or the library. Hence, this strong emphasis on personal responsibility is sometimes too hard and the limits are hard to set. This also requires that the programmers and developers are aware of these ethical questions in general to better recognise ethical questions when they arise.

3 ILLEGAL AND UNETHICAL SOFTWARE

The first question is what is illegal or unethical software. Probably there is no definite answer to this question, since the same code or software is intended to be used in clearly legal purposes, but in some cases, it can be used for illegal or at least unethical situations. In this paper, illegal and unethical software are defined to be software that do not have legal use or they have properties that are unethical from the very beginning - or they can be judged as such, depending on the ethical code of the observer. From this on, unethical systems is used to mean both illegal and unethical systems.

It is clear that any software made to break into systems or to cause harm to the target system or the host computer fall in unethical class. The programmers of these kind of systems are probably well aware of their nature and have made their choice when joined the project. The more interesting cases are cases, where the product itself as a whole is not unethical, but its makers are asked to implement some properties that are not ethical.

3.1 Case Volkswagen emission scandal

In the case of Volkswagen emission scandal, many cars with a Diesel engine had software that was able to decrease emissions when measured in a laboratory. This is probably the most well-known case where unethical software has been made on purpose as a part of legal software. The case is described in more detail in e.g. [6].

The case has an interesting side point, where the software engineers alone were accused for the emission scandal [7]. Later on, this claim was shown false, but even then, this case raises the question if the software engineers should have refused to program such software. According to all of the ethical codes referred in Section 2, they should have done so.

The Volkswagen case is here as a well-known example. It is not known, how much this kind of code exists in general, but it is probably more common than we expect. For instance, there have been several cases when software is suspected to violate the privacy of its users by gathering information of their computer use.

3.2 Code I'm still ashamed of

In 2016, programmers were discussing in Freecodecamp about cases where they have been asked to write unethical code. Results were published in Business Insider (Nordic) [1]. The discussion is still (as spring 2017) online, and can be found in freecodecamp.com, but directly to this discussion it is easier to use reference [8].

One of the main articles on the topic was published by Bill Sourour [2] with a title "Code I'm still ashamed of". In his article, Sourour describes how he as a young professional ended up making a piece of software that - as he feels - caused at least one death. The nature of the software was marketing a drug, and part of it was a quiz, whose results were not used except in the case of allergy, otherwise it suggested the same drug of the client. Only later on, after reading a piece of news where one of the users had made a suicide, the programmer learned that the drug has side effects causing depression and suicidal thoughts. He felt responsible, and concludes:

"As developers, we are often one of the last lines of defence against potentially dangerous and unethical practices."

Other examples in the discussion include a request to use emergency frequencies to make a wireless device work faster, hence endangering emergency messaging if the programmer had not refused. In on other example, a web page had a possibility for the customer opt-out of company newsletters. The company did not honour this, but prepared to send these customers specially edited messages. One of the themes seems to be that when one programmer refused, another one did not.

Anyway, the discussion shows that there are many cases where software professionals are asked to write unethical code. Some of the cases in the discussion are experiences where refusal to write unethical code has succeeded, but a lot of the cases are descriptions where the programmer or somebody else ended to write the code; in many cases, they left the company later on because of these kinds of requests. As in the Sourours article, there are also cases where ethical aspect becomes clear only when it is too late.

3.3 Discussion

There is not much data what would happen, if the programmer refused to write unethical code - in the Freecodecamp's discussion many of the programmers quit the job themselves. It can be asked if they will lose their jobs, but even if not losing their jobs, the refusal to write the code may affect their career, make salary raises smaller or non-existent - things that are hard to show that they originate from the refusal. The problem is that even if there were no consequences, fearing them may affect the programmer. Further, if programmers have a feeling that somebody will do it anyway, this might lower their ethical standard.

In the context of this paper, the cases where ethical implications are found later on are of the most interest. If one was not aware of the ethical point of the view when writing the software, does it free the programmer of the personal responsibility? If it does, does this form an easy excuse? To solve this easy excuse problem, there should be more awareness of ethical questions among software professionals.

4 INCREASING INTELLIGENCE OF THE SYSTEMS

There is a huge number of potential problems that arise with intelligent systems. First of all, who is responsible for the decisions made by the system? Our codes of ethics give responsibility to the engineer, but if the device makes the decision autonomously - especially after a learning process - is the programmer still responsible?

Even if the manufacturer or the vendor is the responsible one, what if the system has been used against the manual, or it has not updated as requested, or something like that falling clearly on the users' or owners' responsibility? Anyway, the owner or user cannot be responsible for decisions they did not have possibility to affect - the manufacturer should be responsible for them. On the other hand, a system that is capable of learning should itself be responsible for the decision - but still, it is a machine. This is one of the main questions to answer before widely used autonomous systems like cars can be taken in a large-scale use.

The responsibility is not the only problem. There are problems especially in cases where there are several possibilities that are almost equal (by some metrics) or all alternatives are bad.

4.1 Biased decisions

When programming a system there are cases where the behaviour is not neutral, like in the example above where the quiz ended up recommending only one drug - the

customer expected to get a neutral answer, but actually got a very biased one. In these kind of questions, the decisions may favour e.g. the customer, vendor, or manufacturer of a product. The customer expects the system work as their benefit prioritised, but the reality may be something else.

4.2 All alternatives are bad

Cases where there are no good alternatives at all are more problematic. Whatever is selected, something bad will take place. If there is acceptable metrics to compare the alternatives, it can be used. Unfortunately, the metrics in these cases is often ethical in nature and may even differ culturally or from the points of view of individuals.

4.2.1 Example of a drowning man and a child

This example come from movie "I, Robot". A detective and a girl are in a danger of drowning. The robot rescued the detective, since he had much better probability to survive compared with the girl who had only 11 percent. This case introduces perfectly logical metrics for selection who to rescue, but the detective himself thought that the robot should have rescued the girl. The scene can be found on Youtube with title "11% Is More Than Enough: Save The Girl".

Computers can handle logical reasoning as in this case. Adding cultural, emotional, or social aspects make decision hard to understand and implement. What is the ethical reasoning that could end up rescuing the girl? Should there be emphasis on the age or sex? Actually, many ethics codes like Archimedean Oath tell us not to let things like age, sex, race or religion affect our decisions.

4.2.2 Example of an autonomous car

Examples of this kind can be found easily from the internet. Consider a case where a completely automatic car is driving on the street and a child runs to the road just ahead of the car. At the same time, a car is approaching on the left line (assuming right-hand traffic), and on the right side there are parked cars.

In any case, the car should try to stop, so the system hits the brakes. Breaking is not enough and three cases can be identified:

1. Avoid collision using the left line.
2. Avoid collision yielding to the right
3. Continue straight.

The first alternative would cause a head-on collision with the approaching car. This alternative endangers passengers in both cars, but saves the child assuming that she will not continue the left line. If the metrics is how many humans are potentially harmed, this may be the worst alternative.

The second alternative would cause a collision with one (or more) of the parked cars. This will endanger the passengers of the car, but saves the passengers of the approaching car and the child - assuming the car will not bounce from the parked car and hit the child anyway. If the autonomous car is empty, and the metrics is the number of potential injuries, this is the best alternative. On the other hand, there may be humans in the parked cars and the autonomous car is not aware of them.

The third alternative is to continue straight and hit the child. Depending on the speed, this potentially kills her, but saves the passengers of the cars.

So, what is the metrics to be used? Does it make any difference if the human is not a child but an adult? What about an animal (a dog or a cat)? In general, if the car has alternatives to save the pedestrian or the passenger of the car, which one it should select? Further, how many people would like to buy a car that does not prioritise the safety of its passengers? Should we make the software ethically configurable so the owner may decide on the metrics?

After all these unanswered questions, a quick look at the possible algorithm may be in place. The algorithm may be something like the following:

1. Something blocks the road.
2. Break!
3. Avoid the obstacle, try left.
4. Left is blocked, too. Try right.
5. Right is blocked, too.
6. Nothing can be done, just try to stop.

As can be seen, the implementer may not be aware any of the ethical questions arising. Even the code making the decision may not indicate the consequences in any way.

5 CONCLUSIONS

For ethical questions, there are no easy answers. In the light of the examples of this paper, one of the problems is to recognise whether the decision has ethical consequences or not. Although not a real case, the case of an autonomous car shows that it is possible to write decision-making software without noticing its ethical consequences. Even in real-world cases this viewpoint was not always seen until later on. Together with strong personal responsibility on the developer of the system, this forms a dire problem.

Particularly in software, where the same piece of code can be reused again and again, and the software is produced as components, the decisions should be identified on the early phases of development. They should not be left for later phases since their nature may not be recognisable. This means the ethical aspects should be evaluated as the software project is started.

In many cases, there are no separate ethical courses for engineers. Having such a course in the beginning of the studies might the waste of time - the students may not be ready to accept that ethical questions are part engineering skills until they understand more about the technology itself. However, ethical aspects should be highlighted in some part of the studies. The best places could be in connection with practical studies like assignments and project courses. For instance, we could require that each assignment or project work in studies include a statement that evaluates if the system have potential ethical questions or not. This way, eventually, the habit would reach the companies, too. As it appears that the answers of ethical questions may be culturally dependent, should we allow the customer to select the underlying ethical principle? If so, the programmers should be even given broader education in ethics.

REFERENCES

[1] Bort, J. (2016), Programmers are having a huge discussion about the unethical and illegal things they've been asked to do. In Business Insider Nordic, 20 Nov 2016. http://nordic.businessinsider.com/programmers-confess-unethical-illegal-tasks-asked-of-them-2016-11. Referred 19.4.2017.

[2] Sourour, B. (2016), The code that I'm still ashamed of. https://medium.freecodecamp.com/@BillSourour. Referred 19.4.2017.

[3] Achimedean Oath (1990). In French: https://www.revolvy.com/main/index.php?s=Archimedean%20Oath. Referred 24.4.2017. Translation in English e.g. in Wikipedia (https://en.wikipedia.org/wiki/Archimedean_Oath), referred 24.2.2017.

[4] ACM (1992) ACM Code of Ethics and Professional Conduct. http://www.acm.org/about-acm/acm-code-of-ethics-and-professional-conduct. Referred 17.4.2017.

[5] IEEE (2006) IEEE Code of Ethics. http://dusk.geo.orst.edu/ethics/codes/IEEE_code.pdf. Referred 17.4.2017.

[6] Volkswagen: The scandal explained, http://www.bbc.com/news/business-34324772. Referred 24.4.2017.

[7] Volkswagen America's CEO blames software engineers for emissions cheating scandal (2015), http://www.theverge.com/2015/10/8/9481651/volkswagen-congressional-hearing-diesel-scandal-fault. Referred 24.4.2017.

[8] Hacker News (2016), Code I'm still ashamed of. (2016) https://news.ycombinator.com/item?id=12965589. Referred 17.4.2017.