Professional Ethics and Codes of Conduct

Mitch Wilson
University of Georgia
CSCI 3030
Instructor: Michael Cotterell

The world we live in today is drowning in technology, but not in a bad way. Once the Internet was invented, the capability of the common user flew off the charts because now anyone could (in the internet's purest form) access any information. This was great for everyone, and it created many fields of study and research that previously had not existed. Computer science and computer engineering are two of the bigger categories that come to mind but there are many others. Since nobody had previously pursued either of those fields before, there were no pre-existing rules governing what was right and what was wrong: the pioneers always got to make their own rules. For most computer scientist, the rules are the code of ethics set forth by the Association for Computing Machinery (ACM).[1] And for most computer engineers that do not fall under the scope of the ACM, there is also a code of ethics set forth by the Institute of Electrical and Electronics Engineers (IEEE).[1] Software engineers have their own code: Software Engineering Code of Ethics and Professional Practice (SECEPP).[1] With so many separate ethical codes, the question arises: is *(insert choice of ethical codes here)* comprehensive enough for the profession for which is was created. For the purpose of this document, the focus will be on the SECEPP, with references to the others. The SECEPP code of ethics is modeled after the IEEE code, where there are listed principles/values that are the broader guidelines and then those values are expanded upon to encompass more specific situations.

One problem spawning from the question of comprehensiveness is that you must first completely define what it is you want to apply to rules to. This can be especially difficult in fields like software engineering. "If it is an entire information system in an organizational context that is the object of interest, then boundary definition becomes an issue. Is the hardware and middleware included? … If however the object of interest, say, a software engineering activity such as testing

[1] Note that in this case, "ACM", "IEEE" and "SECEPP" were used as abbreviations but any further use will be a citation

within a particular project, then the boundary definition becomes easier" (Biffl). In short, this issue can be summarized to mean that as long as we can narrow a situation down one specialization (more so with hardware, or more so with software) then we can more aptly compare the SECEPP ethical code to one either the ACM ethical code (for more software-related cases) or the IEEE ethical code (for more hardware-related cases). Also, in any instance where there is a problem created by the computer there are two options for what caused the fault: either from a hardware or software malfunction, or from misuse by the user. Any case where a malfunction occurred could be considered a breach of ethical conduct as the SECEPP has multiple listings the ethical code against delivering a faulty product: "Ensure adequate testing, debugging, and review of software" (SECEPP 3.10).

I must point out that misuse by the user is not a breach of ethics by the engineer. Anyone who has taken a coding class can tell you that you cannot predict every possible way the user could misuse the program. Kantianism, in its rudimentary form, states, "if one person is allowed to do something, then all peoples will be allowed the same." Applying this to a situation where a user attempts to "hack" a program and then extrapolating using Kantianism would create a society where every user hacks the program, which would be an unethical world, and therefore it would be unethical for a single person to attempt to hack the program. In a case such as this, adequate measures to ensure proper use would suffice to show adherence to the code. And, as proven, a broader school of ethics can be applied to defend the engineer's adherence, assuming that they did in fact "perform adequate testing" and made a reasonable effort to "maintain the integrity of [the program's] data" (SECEPP 3.10, 3.14).

In September 2015, it was discovered that certain Volkswagen cars had software that could detect when the car was being emissions tested, and then alter the performance of the vehicle so the emissions test would be passed. "When the cars were tested under laboratory conditions…the [software] put the vehicle into a safety mode in which the engine ran below normal power and performance. Once on the road, the [software] switched the engine back to full power" (Hotten). According to the article, this "two-faced" engine and software combination led to the vehicles passing emissions inspections and then releasing up to 40 times the legal limit of nitrogen oxide once operating back at full power. This software and its creation is in direct violation of the SECEPP code of ethics, mainly the first virtue which relates to operating in the public interest. In particular: "Approve software only if they have a well-founded belief that it is safe, meets specifications, passes appropriate tests, and **does not** diminish quality of life, diminish privacy or **harm the environment**. The ultimate effect of the work should be to the public good" (SECEPP 1.03). When the engineers knew what the software would be used for they knew that to continue work would be against the code of ethics. Applying Kantianism to this situation would create an individual rule saying that it is okay for a team of engineers to violate the code of ethics because it was their job. This would create a universal rule where all software engineers could violate the ethical code if it was what was assigned. This would negate the purpose of even having an ethical code and would create an environment of chaos that would be detrimental to the field.

The benefit of having a code of ethics is only seen when the code is upheld by those whom it covers, and when it is upheld it provides a benchmark for the rest of the community. A problem arises when the code is written to be all encompassing and it ends up contradicting itself in a situation. That situation applies to the Volkswagen case. It is clear that the team who created the software violated an entire section of the ethical code, but it could be argued that they were in compliance with part of the code, which states "accept no work detrimental to the work they perform for their employer" (SECEPP 2.08). In this situation, it would be against the code for the engineers to not create the software, because not creating the software would be detrimental to the company. It is a long shot to make this argument but still one that could be made if the case went to court. Although this contradiction is vague, it still highlights one of the shortcomings of a code designed to cover many facets of a certain field.

It is still in the best interest of every party for there to be a written code of ethics, at the very least for the lone soul to quote in their defense of an ethical act, regardless of how encompassing or simple the code is. However, as demonstrated by the Volkswagen case, any code of ethics is negated when the engineers don't uphold it.

BIBLIOGRAPHY

"ACM Code of Ethics and Professional Conduct." ACM Code of Ethics and Professional Conduct. ACM. Web. 27 Apr. 2016. Adopted 16 Oct 1992.

"Association for Computing Machinery." Software Engineering Code of Ethics and Professional Practice —. Web. 27 Apr. 2016. Adopted 1999.

Biffl, Stefan. Value-Based Software Engineering. Springer Science and Business Media, 2006. Google Books. Web. 28 Apr. 2016.

Hotten, Russell. "Volkswagen: The Scandal Explained - BBC News." BBC News. 10 Dec. 2015. Web. 29 Apr. 2016.

"IEEE Code of Ethics." IEEE. IEEE. Web. 27 Apr. 2016.