About Integrating Ethics in The Software Engineering Curriculum

Zainab KHALLOUF, Ph.D.
Faculty of Information Technology
Homs-Syria
Tel : 0966176392
zainab.khallouf@gmail.com

Professionalism and accountability, intellectual property rights, privacy and information protection, and a variety of other ethical dilemmas are an integral part of the software engineer’s eventual career. The software engineering community has therefore recognized the importance of building a culture of software engineering ethics to guide the behaviour of and decisions made by professional software engineers to ensure, as much as possible, that their efforts will be used to make software engineering a beneficial and respected profession. However, the lack of consensus regarding the teaching methodology, the tensions that arise when adhering to different social or legal standards, and sometimes the lack of interest and experience, hinders the effective integration of ethics in the academic programmes. This article discusses these challenges, and presents some approaches to teach the software engineering ethics.

Keywords: Computer ethics, software profession, professional ethics, curriculum studies, computer science education.

1 Introduction

Facing Ethical dilemmas is an integral part of the software engineer’s eventual career. The software engineering community has therefore recognized the importance of building a culture of software engineering ethics to guide the behaviour of and decisions made by professional software engineers to ensure, as much as possible, that their efforts will be used to make software engineering a beneficial and respected profession. Furthermore, there has been a consensus of integration of ethics within Software Engineering curriculum. From this perspective, the joint IEEE-CS/ACM Software Engineering Coordinating Committee (SWECC) was formed, in 1998, to act as a permanent entity to foster the evolution of software engineering as a professional computing discipline. The committee was created to oversee three initiatives: the development of curriculum guidelines for software engineering, the definition of

SCS workshop 2007
a code of ethics and standards of professional conduct for software engineers and to the Software Engineering Body of Knowledge (SWEBOK) Guide project [3].

The ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices introduced a code of ethics (sometimes called a code of conduct or code of practice) for the software engineers. This Code contains eight principles related to the behaviour of and decisions made by professional software engineers. In addition, the codes represent a movement toward an international consensus of what software engineers believe to be their professional ethical obligations [1]. Nevertheless, in some situations standards may be in tension with each other or with standards from other sources. These situations require the software engineer to use ethical judgment to act in a manner which is most consistent with the spirit of the Code of Ethics and Professional Practice, given the circumstances. That is, reaching an ethical decision in these cases needs a thorough thinking rather than blind adhering to regulations [1, 7, 8]. This article looks at issues like: Can the code guide the software engineers in their decisions making in all the circumstances? Do we need to introduce the student to the ethical theories and the basis of ethical thinking before introducing the code? What is it the best methodology to teach ethics? What could a syllabus for an ethical course look like?

The remainder of the paper commences with an overview of the Software Engineering body of knowledge and the professionalization of software engineering. This is then followed by an exposition of the ethical theories and their rationale. Next, section five, discusses the difference between using codes of ethics or the ethical analysis to take a decision. Section six presents a proposal for a syllabus for software engineering ethics course. And section seven concludes this paper.

2 The Software Engineering Body of Knowledge (SWEBOK)

Generally speaking, the software engineering body of knowledge characterized what knowledge and capabilities a good Software Engineer must have. That is, the software engineering body of knowledge provides a consensually validated characterization of the bounds of the software engineering discipline and to provide a topical access to the Body of Knowledge supporting that discipline. During the late 1990s, the ACM and the IEEE organized many academics and practitioners into several SWEBOK (Software Engineering Body of Knowledge) committees. The main published work of this research is the 2004 Version of the SWEBOK Guide [3]. The guide defines ten Knowledge Areas (KA).

- Software requirements.
- Software design.
• Software construction.
• Software testing.
• Software maintenance
• Software configuration management.
• Software engineering management.
• Software engineering process.
• Software engineering tools and methods.
• Software quality.

The guide places ethics in the Software Quality KA as what is called the Software Engineering Culture and Ethics. So, ethics plays a significant role in software quality, the culture, and the attitudes of software engineers.

3 The Professionalization of Software Engineering

The professionalization of software engineering implies transforming the software engineering to a profession with a standard underlying set of principles and concepts. (Ford et al.1996) [4] represents the maturity of the software engineering profession by a model composed of eight infrastructure components, these components are:

• Initial Professional Education.
• Accreditation.
• Skills Development.
• Certification.
• Licensing.
• Professional Development.
• Code of Ethics.
• Professional Society.

Figure 1 shows the infrastructure-level components of a profession and suggests a typical path for a person choosing to enter that profession [4]. The aspiring professional first undertakes initial professional; the quality of a professional degree program is assured by accreditation. To become a professional, he or she must

SCS workshop 2007
develop skill in the application of that education (through university co-op pro-
grams, on-the-job training, apprenticeships, internships, or other means). Certification
and/or licensing assure the competence of the individual to enter professional
practice. Throughout practice, there are periods of professional development, possibly
resulting in re-certification or re-licensing. The profession assures that its prac-
titioners behave in a responsible manner by defining a code of ethics. A professional
society helps assure that all the other components interact appropriately.

To sum up, ethical behaviour should be taught whenever a profession wants to
be established.

4 A Brief Look at Ethical Theories [6]

Computer ethics is the study of the ethical questions that arise as a consequence of
the development and deployment of computers and computer technology [5]. Or, as
Baase (1997) simply says: Ethics is the study of what it means to do the right thing.

4.1 Examining a Case for Ethical Issues

There are several steps to make and justify ethical choices, these steps are:

- Understand the situation. Learn the facts of the situation. Ask questions
  of interpretation or clarification. Attempt to find out whether any relevant
  forces have not been considered.

SCS workshop 2007
• Know several theories of ethical reasoning. To make an ethical choice, you have to know how those choices can be justified.

• List the ethical principles involved. What different philosophies could be applied in this case? Do any of these include others?

• Determine which principles outweigh others. This is a subjective evaluation. It often involves extending a principle to a logical conclusion or determining cases in which one principle clearly supersedes another.

4.2 Examples of Ethical Principles

There are two different schools of ethical reasoning: one based on the good that results from actions (i.e. Consequence-Based Principles), and one based on certain prima facie duties of people (i.e. Rules-based principles).

• The Teleological Theory:
  This kind of ethical reasoning focuses on the consequences of an action. The action to be chosen is that which results in the greatest future good and the least harm. There are two important forms of teleology. Egoism is the form that says a moral judgment is based on the positive benefits to the person taking the action. The other principle is the utilitarianism. In this situation, the utilitarian would assess personal good and bad, good and bad for the company, good and bad for the customer, and, perhaps, good and bad for society at large.

• The Deontology Theory:
  The deontology is founded in a sense of duty. This ethical principle states that certain things are good in and of themselves. Examples of intrinsically good things:
  – Truth, knowledge, and true opinion of various kinds, understanding, and wisdom.

5 Codes of Ethics vs. Ethical Thinking

To understand the differences between the codes of ethics and ethical thinking, let’s discuss two cases. The first one, deals with the responsibility of the software engineer toward the public, specifically, the case discusses when the software engineer has to blow the whistle for an action he (she) thinks improper. The second case discusses the ethics behind the free software as presented in an article by Richard Stallman, the founder of GNU organization.
• Case 1 [9]:

*Suppose a programmer discovers that a software product he/she has been working on is about to be released for sale to the public even though the product is unreliable because it contains "buggy" software, should he/she blow the whistle?*

– Using the Codes of Ethics (Appendix A)

The case is related to the two following rules:

6.08. Take responsibility for detecting, correcting, and reporting errors in software and associated documents on which they work.

Principle 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.

So the programmer in this case has to blow the whistle.

– Using the ethical theories

From the *Egoism’s view*: An egoist weighs the outcomes of all possible acts and chooses the one that produces the most personal good for him or her with the least negative consequence. The effects on other people are not relevant. So the programmer does not care, he/she might argue as follows. "If I neglect this action, I will satisfy my manager, my company’s reputation may be tarnished, but that will not be tracked directly to me. Thus, I can justify writing shoddy code."

In contrast, the *utilitarian* chooses that action that will bring the greatest collective good for all people with the least possible negative for all. In this situation, utilitarian would assess personal good and bad, good and bad for the company, good and bad for the customer, and, perhaps, good and bad for society at large. So under this principle, this programmer has to blow the whistle to save the reputation of the company and to protect the final users.

Now, the *deontology* principle states that certain things are good in and of themselves. These things that are naturally good are good rules or acts, which require no higher justification. Something just is good; it does not have to be judged for its effect. Moreover, certain universal, self-evident, natural rules specify our proper conduct. Certain basic moral principles are adhered to because of our responsibilities to one another; these principles are often stated as rights: the right to know, the right to privacy, the right to fair compensation for work. Nonmaleficence, is one of these rules. So under the concern of not harming others, the programmer must blow the whistle in particular if he/she knows that the released software could have catastrophic results on the users. The last

*SCS workshop 2007*
case is not easy in practice, the possible harm is being fired. However, he/she has a potential gain by being able to "blow the whistle".

- Case 2 [10]

*Should Software be free?*

Richard Stallman explains why software should be free, using the following rational: Teleological analysis: Compare the value to society of a restricted (i.e. proprietary) program with that of the same program, available to everyone, assuming development costs have been paid.

- Fewer people use the program.
- None of the users can adapt or fix the program
- Other developers cannot learn from the program, or base new work on it.

Levels of psychological harm:

- Damage to social cohesion: ethically (and politely), if I have a program and my neighbor wants to use it, I should share. Signing a typical software license agreement means betraying your neighbor.
- Damage to the programmer: many users will not be allowed to use the programmer’s work which leads to a general attitude of cynicism.
- Frustration at having to accept a system that does not fit a user’s needs
- Damage to the spirit of scientific cooperation: nowadays only enough information is given to allow a programmer/researcher to marvel at what was done, but never enough for someone to recreate it.

In the light of the previous analysis, we have the feeling that codes alone can not help to justify all choice, as we see in the rational of free software. So, going beyond the code is needed to build a healthy ethical education.

6 A Proposal for the Computer Ethics Course Syllabus

- Introduction to computer ethics: Terminology, definitions, history, and field of application.
- Ethical thinking: Ethical theories, Code of ethics.
- Case studies: Privacy, confidentiality, intellectual property, Internet Ethics and Freedom of expression.

This part focuses on presenting some situations related to the daily use of

*SCS workshop 2007*
Information and networks, and discussing these cases in the light of ethical theories. We can discuss issues like copyright, publishing on the Internet, and freedom of expression.

- Open and Free software.
  The ethics course presents a good opportunity to introduce the philosophy behind the open and free software.

- The ethics of computer forensics.
  Computer forensics is the field of study that deals with finding the digital evidences and incidences response. From our teaching experience, dealing with such a subject in the ethic course could motivated the students by making them discover a new field and in the same time use this new field as a tool for presenting some ethical principles.

- Presentations of papers chosen by the students.

7 Conclusion

The software engineering ethical principles assure that the behaviour of and decisions made by professional software engineers to ensure, as much as possible, that their efforts will be used to make software engineering a beneficial and respected profession. A general feeling is that teaching ethics needs to go beyond the code and move toward ethical analysis in general. Ethical theories and case studies are a good beginning to make the students appreciate the field and make the ethics course a rewarding experience. Ethics is a way to introduces human factors in software development and an effective step toward establishing a healthy software engineering profession.

References


SCS workshop 2007
Appendix A. Software Engineering Code of Ethics and Professional Practice (Short Version)

Software Engineering Code of Ethics and Professional Practice (Version 5.2) as recommended by the ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices and jointly approved by the ACM and the IEEE-CS as the standard for teaching and practicing software engineering. Short Version

PREAMBLE
The short version of the code summarizes aspirations at a high level of the abstraction; the clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code. 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:

- PUBLIC - Software engineers shall act consistently with the public interest.
- 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.
- PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
- JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.
- MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

SCS workshop 2007
• PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

• COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.

• 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.