Professional and Ethical Dilemmas in Software Engineering: Does the ACM Software Engineering Code of Ethics Need Revision?

Brian Berenbach
Siemens Corporate Research, Inc.
brian.berenbach@siemens.com

Manfred Broy
Technische Universität München
broy@in.tum.de

Abstract

A name is a powerful thing. Naming a disease, for example allows doctors to identify and share information about it, hopefully leading to a cure, and even better, prevention. In the field of philosophy, certain behaviors are named to better identify them, e.g. "bandwagon effect". In software engineering, no such names exist for unprofessional behavior or ethical dilemmas. Even worse, practitioners sometimes engage in unprofessional or unethical behavior without realizing it. Unethical and unprofessional actions can perturb the software lifecycle, resulting in additional downstream unethical or unprofessional behavior. Propagated unprofessional or unethical behavior may occasionally have catastrophic consequences. In this paper, we identify, categorize and name nine specific ethical and professional dilemmas in software engineering with the hope that giving such behavior a name will increase awareness and decrease the frequency with which they occur. These dilemmas are placed in the context of the ACM code of conduct.

1. Introduction

We often engage in unprofessional or unethical behavior without realizing it [7]. In ethics courses, or through professional association codes, practitioners may be warned of situations or dilemmas that may eventually lead to unethical behavior, e.g. "Avoid harm to others". The joint ACM/IEEE-CS task force has created a code of ethics for addressing issues of unethical behavior. [1]. But too often the focus is on headline scenarios and not on the [initially] mundane situations that abound in our profession. Furthermore, ethical training may not be given the needed emphasis during undergraduate education [3] [6] [16].

What typically makes the headlines is not a small ethical lapse, but rather the result of a sequence of related ethical lapses. When such scenarios cascade, there tends to be a magnification effect (see section 6).

An ethical dilemma occurs in software engineering when the professional must make a choice between competing values, e.g. personal vs. professional. For example, a sales manager may sign a contract to deliver a software product knowing, or having been advised, that the product will take longer to deliver than the promised date. The sales manager's dilemma might be that he is under pressure to meet a financial target or there might be job-related consequences. The project manager then creates an unrealistic project schedule, with lead development staff choosing to defer for political or organizational goals. A chain of unethical or unprofessional behaviors then may take place that may eventually lead to massive penalty payments, lawsuits, layoffs, or even bankruptcy. Yet none of the players in this process will admit to or recognize that anything in his or her behavior was unethical or unprofessional.

To shine some light on these kinds of subtle but nonetheless unethical or unprofessional situations we have taken the bold step of giving each dilemma a name. The reader may, of course, disagree with our choice of labels, but we doubt whether there will be disagreement as to whether the behavior is inappropriate.

2. Unprofessional vs. Unethical Behavior

Not every wrong behavior is unethical. If people do not know better and behave wrongly, they are not unethical. It certainly is unethical, however, if decisions are taken by people that know that they do not know enough about a field, in particular, to do professional decisions. The term "ethical behavior" refers to how an individual or an organization ensures that all its decisions, actions, and stakeholder interactions conform to the individual's or the organization's moral and professional principles. These principles should support all applicable laws and regulations and are the foundation for the individual's or the organization's culture and values. They define right from wrong.

3. Categorization of dilemmas

The situations that make the headlines are typically brought on by incompetence, unprofessional behavior, personal misconduct, mismanagement or more commonly, by a seemingly inconsequential chain of small ethical lapses.

For example, if a project is running late, the project manager may be tempted to cut short the requirements definition phase in the hope that the time can be made up later. Software developers will then have incomplete requirements and, under pressure to produce, fill in the missing pieces. In order to get the product out the door, testing is based not on the requirements, but on developer descriptions of how their code will work. The result is then delivered to the customer with possibly

catastrophic consequences, e.g. unusable product, contract cancellation, lawsuits, etc. Such a behavior is at least shortsighted and thus not professional. Wrong decisions lead to bad results. If the wrong decisions are taken by a person that knows better and if the decisions are motivated by personal interests, then the behavior becomes unethical.

When students are introduced to ethics courses they learn about situations that are clear and extreme. It is relatively easy to distinguish under such circumstances when behavior crosses the line or is unethical. However, real life is not so simple, and the dilemmas described below come from scenarios that occur all too frequently.

3.1. Mission Impossible

The **Mission Impossible** dilemma occurs when an individual is asked to create or accept a schedule that he or she knows to be impossible to meet. Because of felt political pressure or for other reasons the person creates or accepts the schedule knowing that it is not realistic.

The consequences of this lapse of judgment can range from loss of qualified staff to significant loss of revenue. Loss of staff comes from overwork and burnout. Loss of revenue could derive from the premature announcement of the availability of a product that is then delayed in the marketplace: customers stop buying the current product in anticipation of the new product arriving.

3.2. Mea Culpa

This dilemma occurs when staff members are asked to deliver product while it still is missing functionality or has known software defects.

The pressure to release prematurely can come from market pressure, i.e., e.g. get it out the door before the competitor does or contractual obligations, possibly associated with a penalty clause.

A risk assessment is worthwhile; it could be of benefit to release early under certain circumstances. However, problems may arise when no risk assessment is done or the actual risks are much greater than the perceived risks.

The short term result of the delivery of incomplete software products is customer dissatisfaction. Long term repercussions may include a bad reputation and loss of market share or sales. If incomplete deliveries happen often enough, the company may go out of business. In a worst case scenario, company staff could expose itself to civil or criminal penalties see section 4 below.

3.3. Rush Job

Occasions can arise in which quality is compromised because of either poor work ethic or perceived pressure to deliver. A developer working on a software product delivers working code, but the quality of the work product is shoddy, with minimal or no annotation, and little or no documentation. The programmer may feel under pressure to deliver, and is more concerned about meeting milestones than quality.

The **Rush Job** and **Mea Culpa** dilemmas are quite different. While in Mea Culpa a product missing functionality is being delivered, in the **Rush Job** scenario, full functionality may be present, but the product is of low quality, e.g. does not meet quality standards as a result of intentionally trading quality for speed of implementation.

3.4. Not My Problem

On occasion, a project team or staff will concern itself with day to day activities, accepting the development culture status quo and not seeking to improve productivity or quality. For example, error codes might be hard coded in the software rather than being placed in a table. When best practices are ignored, the door may be opened for civil, and in some rare cases criminal, liability. We have called this dilemma **Not My Problem** because of the frequency with which team members will state that quality, productivity and best practice issues are someone else's responsibility.

3.5. Red Lies

Red Lies occur during meetings with clients or management where statements about product or project are made that are known to be untrue. An example would be to state that delivery is on schedule when it is common knowledge on a team that it is not feasible to deliver on time. There is a little bit of **Not my problem** in **Red Lies**. For example, rather than admit that a project is behind schedule, a project manager might rationalize that it is the responsibility of the development team to make up the lost time, and, consequently, an idealized schedule can be reported; if the development team does not make their schedule it is their problem. The use of the color red is indicative of what may happen to the company bottom line if this behavior is pervasive or ongoing.

3.6. Fictionware

Fictionware occurs when an organization or individual promises or contracts to deliver a system for which some agreed on features are not feasible. The difference between **Fictionware** and the frequently used term "Vaporware" is that in **Fictionware** a product exists but does not have the specified functionality. In the case of "Vaporware" the product simply does not exist at all. This situation typically occurs when people feel under intense pressure to meet sales targets or to get

the big one; denial can cause more difficulty reading a request for proposal than cataracts.

Fictionware contracts are endemic in contracting organizations where sales commissions are decoupled from delivery. The sales representative may have only a vague understanding that the contracted for project is not feasible, but really doesn't care because his or her commission is only contingent on contract award, and not based on long term profitability. The best way to mitigate problems of **Fictionware** is to couple merit or bonus payments to after completion project profit, and to give engineering professionals significant up-front responsibility and authority to influence the bidding or quotation process.

3.7. Non-diligence

Non-diligence occurs when important documentation such as requests for proposals, requirements documents or contracts are not given a thorough review. **Non-diligence** is a bit different from **Fictionware**. With **Fictionware** there is acceptance of the responsibility to deliver something that is not possible. In the case of **Non-diligence**, agreements may be made without a careful understanding of what is being agreed to because of either failure to carefully evaluate a specification, or failure to pay close attention to staff when they voice their concerns.

3.8. Cancelled Vacation Syndrome

A **Cancelled Vacation Syndrome** occurs when managers pressure staff members at the last minute to cancel planned trips or otherwise sacrifice their personal time and possibly money, e.g. trip reservations, in order to meet a short term deadline.

While working at a consulting company, one of the authors observed several consultants being told to cancel their vacation plans so that a project milestone could be met. In one such case, the employee's parents were flying in from overseas and the trip plans had been finalized nearly a year before the trip date. The forced cancellation indicated a lack of planning on the part of project management, and while potentially solving a short term problem, in this particular situation caused an even more serious long term staffing and morale problem. Every employee who was asked to cancel his or her vacation left the company within a year. Moreover the project was cancelled shortly after the trip cancellations occurred. So the company that fostered the **Cancelled Vacation Syndrome** gained nothing and lost several valuable employees.

3.9. Sweep It Under The Rug Syndrome

The **Sweep It Under The Rug Syndrome** occurs when unforeseen issues arise that could potentially damage a project or company, but to keep things running smoothly, the issues are ignored in the futile hope that they will vanish. For example, a tester uncovers a flaw in a communication system and calls it to the attention of his supervisors. They determine that while the flaw is a real problem, the odds of it impacting the delivered product are relatively small, and besides, once the customer starts using it, the responsible parties will have moved on to another project. **Sweep It Under The Rug** differs from **Not my problem** in that it deals with the mishandling or ignoring of infrequently occurring problems of a unique nature, whereas **Not My Problem** occurs where there are systemic problems in infra-structure or process that are not addressed.

4. The ACM Code of Ethics

The ACM code of ethics contains a set of 24 imperatives that deal with professionalism, the interaction between professionals and society, and leadership [1] [13] [17]. We have cross referenced the dilemmas listed above to their relevant imperatives in the ACM code (Table 1). The imperatives are well crafted and comprehensive. However, they have not served the software professional community as well as they might for a variety of reasons:

- A large percentage of software professionals do not belong to the IEEE or ACM.
- Many individuals working on projects may not be software professionals, e.g. he or she may be product manager or project manager.
- Many ACM and IEEE members are unfamiliar with the society code of ethics.
- Even when somewhat familiar with the imperatives, peer, organizational or other pressures may be brought to bear.
- The imperatives in some cases are vague and require study to understand when they apply to a particular situation.

We have selected several of the imperatives that are relevant to the ethical dilemmas described above to highlight how the imperatives may not provide adequate guidance to the software professional during daily activities.

4.1. Be honest and trustworthy

The question of what honesty entails may be open to question. Just as a heavy gravitational field can bend light, heavy organizational or financial pressure can bend the truth. For example:

- telling a client that software is operational when, in fact, it is under construction,
- forecasting a delivery date that is achievable only if the staff works 24-hour days,

Table 1 Cross reference of Dilemmas with Imperatives

Ethical Dilemma	Applicable ACM Imperatives	Comment
Mission Impossible	Honor contracts, agreements and assigned responsibilities – "a computing professional has a responsibility to request a change in any assignment that he or she feels cannot be completed as defined"	The difficulty with honoring agreements and not accepting impossible assignments is that often the organizational culture is one where acceptance of any assignment is the norm where the assignment is coming from a supervisor.
Mea Culpa	Strive to achieve the highest quality, effectiveness and dignity in both the process and products of professional work – "The computing professional must strive to achieve quality and to be cognizant of the serious negative consequences that may result from a poor quality system"	The imperative is too broad to allow the professional to recognize when it applies in routine situations.
Rush job	See Mea Culpa	See Mea Culpa
Not my problem	See Mea Culpa	See Mea Culpa
Non-diligence	Give comprehensive and thorough evaluations of computer systems and their impacts, including possible risks.	Mixed teams of project management, marketing and sales may make it difficult to achieve this objective, especially if the opinions given do not coincide with the goals of senior management.
Fictionware	Be honest and trustworthy	Honesty and trustworthy are much more difficult to achieve with organizational dynamics than as an individual. Nonetheless, per the ACM imperatives, there are times when a professional should take a stand or walk away from an assignment.
Cancelled Vacation Syndrome	Not covered by the ACM code of ethics. The ACM imperatives deal with fairness and discrimination, not the mistreatment of staff.	The ACM code only deals with generic fairness and non-discrimination.
Sweep it under the rug syndrome	Strive to achieve the highest quality, effectiveness and dignity in both the process and products of professional work. Also, honor contracts…	Problems that occur during construction and testing of software are often resolved by management. Unfortunately, many managers are unaware of or consider themselves not bound by ACM ethical codes.

- stating that there are no known problems with software when, in fact, serious problems have been reported by testing or development.

The problem is not only that honesty may be open to interpretation, but that there may be intense organizational and financial pressure being applied to see issues with a particular slant.

4.2. Strive to achieve the highest quality, effectiveness and dignity in both the process and products of professional work

We learn in requirements engineering that terms like "highest quality" are not quantifiable and are inherently ambiguous. We also know that quality comes at a price. When reducing defects in a product, there is a point where the cost to find the last few defects outweighs the benefit of finding them. The issue is sometimes that recognizing that achieving the highest quality may not be feasible renders the whole issue of quality moot, e.g. throwing the baby out with the bath water.

For example, because of a shortage of professional staff or for other reasons, there are no peer reviews on a project. Code reviews are one of the most effective mechanisms for finding software defects; without peer reviews, an organization may be asking for trouble. Staff may not have awareness that reviews are missing from their process, e.g. business as usual; they may recognize that the reviews are missing but accept their management's position that there is no time for code reviews.

If an organization is not diligent, its process can easily degenerate into an anarchic environment of hacking.

5. Criminal vs. Unethical Behavior.

There are occasions when an individual or organization may engage in unethical practices that are criminal in nature [12]. The individual may not realize that the practice or lack of best practice is criminal. Note that in the continental United States, every state may have variance in how the laws are interpreted. Outside of North America, laws may vary widely, and a practice that is not criminal in one locale may, in fact, be criminal in another. The purpose of this section of the paper is to show how individuals may break the law without realizing it. In all cases at all times, software professionals should be cognizant of the financial, legal and political repercussions of irresponsible behavior, including condoning behavior that they themselves may or may not be part of.

5.1. Negligent Homicide

Negligent Homicide is the killing of another person through gross negligence or without malice. An unintentional killing in which the actor should have known they were creating substantial and unjustified risks of death by conduct that grossly deviated from ordinary care summarizes the relationship between the definitions of these terms [18].

5.2. Reckless Endangerment

A person is guilty of reckless endangerment when, that person recklessly engages in conduct which creates a grave risk of death of another. Note that the person engaging in the behavior may not be aware that they are endangering the welfare of other people. "Reckless" behavior is sufficient.

5.3. Depraved Indifference

Depraved Indifference can, in some situations, depending on the laws of the jurisdiction, be considered a more serious kind of reckless endangerment where the person engaging in behavior that took the life of another did so under circumstances evincing a depraved indifference to human life. That is, he or she was fully aware that his or her actions might lead to injury or death, but was indifferent to that outcome.

5.4. The relationship of unethical to criminal behavior.

As stated above, a professional may engage in criminal behavior without realizing that he or she is doing it. For example:

- Not tracing requirements to test cases. For example, a requirement may state that a radiation dosage for an x-ray machine can, under no circumstances exceed 5 RAD. If the developer permits larger doses to occur and the tests do not pick this up, the organization may be criminally liable for failure to follow best practices, e.g. end-to-end traceability.
- Hard coding error codes. On a rail signaling system the errors are hard coded. Without a table containing the error codes, it is not possible to effectively find and test each error condition. During operation, an untested error occurs that causes a signal to freeze in the go or green position resulting in a train collision. The company that created the signaling system can be held criminally liable because it can be shown in basic software engineering texts that error management during software development is a fundamental best practice.

One of the ACM imperatives is "Know and respect existing laws pertaining to professional work"**.** Unfortunately, there are some infrequent situations where professionals learn **after the fact** that they may have broken laws or may be subject to criminal or civil penalties. In the eyes of the law, it is still the individual's responsibility to ensure that best practices are followed, *regardless of perceived organizational or management pressure*.

6. Dilemma Magnification Effects

When ethical dilemmas are coupled or chained together the results may be more damaging that any one of the dilemmas occurring by itself [2]. For example, **Non-diligence** may result in late delivery**.** At that time, it might be possible to renegotiate a viable schedule. However, in order to make up for lost time, **Mission Impossible** results in difficult schedules with organization pressure applied. The probability of project failure has now increased as things become more chaotic and process suffers. Delivery pressure then results in the **Rush Job** dilemma as developers abandon best practices to get software out the door. The probability of project failure has just increased again, as failure to follow best practices may result in an untenable product or delivery that never works.

In general, the dilemmas tend to cascade, where the earlier in the process the dilemmas are recognized; the easier it may be to alter behavior and steer towards a positive or less negative outcome.

7. Mitigation Strategies

When faced with a potential ethical dilemma, one of the best mitigation strategies is to perform a risk analysis before deciding on a course of action. One of the best preventive strategies is to have a working code of conduct and conduct ethics training for all staff. Some prevention strategies are shown in the mitigation table below:

Ethical Dilemma	Possible Mitigation Strategy
Mission Impossible	Risk analysis. Inform management of the potential long term consequences of not accepting a viable schedule
Mea Culpa	Risk assessment. Good customer management
Rush Job	Enforced quality standards. Staff training.
Not my Problem	Special training for individuals in technical management positions. Open communication channels. Staff round-table discussions.
Non-diligence	Training of staff prior to assignments. Adherence to code of conduct. A professional atmosphere. Coupling of bonuses and incentive pay to overall profitability.
Fictionware	Same as non-diligence
Cancelled Vacation Syndrome	Avoidance of **Mission Impossible**. Holistic risk assessment of alienating staff. Review and correction of problems with estimation.
Sweep It Under the Rug Syndrome	Same as for **Not my Problem**. Conducting a risk assessment is usually worthwhile.

8. Conclusions

While most companies and organizations have ethical codes of conduct, software professionals may not recognize that the codes apply to every day practices. We have presented and named nine ethical dilemmas that commonly occur during software or mechatronic product development. All of the dilemmas described in this paper are common occurances, often the participants don't recognize that their behavior is unethical [8]. Perhaps by naming the dilemmas, as with software patterns, it will be easier to recognize their occurrence and take corrective action.

Ethical dilemmas can cascade, with an increased probability of project failure with each misstep. Unfortunately, the data are not available to quantify the contribution of each dilemma to the probability of project failure [4]; it is usually buried in the failed projects file cabinet.

One possible mechanism for preventing such behavior is professional or corporate education [9]. Clearly, it is not enough to reach out to members of the IEEE or ACM as the initial or causal dilemma in a chain may occur further upstream, e.g. during contract negotiation or product definition.

Furthermore, the IEEE or ACM ethical imperatives need to be communicated to computer science and software engineering students and professionals in a down to earth way so that they can recognize unethical behavior, e.g. see the relevance to their work, and stop or mitigate it in a timely manner [5][10][11][14].

9. References

[1] http://www.acm.org/about/code-of-ethics

[2] R. P. Feynman, "Appendix to the Roger's Commission Report on the Space Shuttle Challenger Accident", Report of the Presidential Commission on the Space Shuttle Challenger Accident, Washington, D.C. , June 6th, 1986

[3] V. Flynn and T. Hall, "Ethical Issues in Software Engineering Research: A Survey of Current Practice", Empirical Software Engineering, vol. 6, no. 4, pp. 305 – 317, Dec. 2001.

[4] J. Singer and G. Vinson, "Ethical Issues in Empirical Studies of Software Engineering", IEEE Transactions on Software Engineering, vol. 28, no. 12, pp. 1171-1180, Dec. 2002.

[5] D. Gotterbarn, "Ethical Considerations in Software Engineering", in Proceedings of the 13th international conference on Software engineering, 1991, pp. 266-274.

[6] E. Towell, "Teaching Ethics in the Software Engineering Curriculum", in Proceedings of the 16th Conference on Software Engineering Education and Training, 2003, pp. 150-157.

[7] T. Forester and P. Morrison, “Computer Ethics: Cautionary Tales and Ethical Dilemmas in Computing”, Harvard Journal of Law & Technology, vol. 4, Spring Issue, pp. 300-305, Spring 1991.
[8] K. Brunnstein, “Why a Discussion on Ethical Issues in Software Engineering is Overdue”, in Ethics of computing: codes, spaces for discussion and law, London: Chapman & Hall, Ltd, 1996, pp. 52-55.
[9] H. Pournaghshband and A. Salehnia, “Ethical Issues in Software Engineering Revisited”, in IRMA International Conference, 2001, pp. 253-256
[10] D. Gotterbarn, “Raising the Bar: a Software Engineering Code of Ethics and Professional Practice”, in Proceedings of the ethics and social impact component on Shaping policy in the information age, New York: ACM Press, 1998, pp. 26-28.
[11] J.B. Thompson and E. Towell, “A Further Exploration of Teaching Ethics in the Software Engineering Curriculum”, in Proceedings of the 17th Conference on Software Engineering Education and Training, 2004, pp. 39-44.
[12] J.R. Herkert and B.M. O’Connell, “Teaching Product Liability as an Ethical Issue in Engineering and Computer Science”, in 33rd ASEE/IEEE Frontiers in Education Conference, 2003, pp. S2A-12.
[13] D. Gotterbarn, K. Miller, and S. Rogerson, “Software Engineering Code of Ethics”, Communications of the ACM, vol. 40, no. 11, pp. 110-118, Nov. 1997.
[14] E. Georgiadou and P.K. Oriogun, “Professional Issues in Software Engineering Curricula: Case Studies on Ethical Decision Making”, in International Symposium on Technology and Society, 2001, pp. 252-261.
[15] K.M. Passino, “Ethical Dilemmas, Choices, and Codes of Ethics”, presented at Professional Aspects of Electrical and Computer Engineering, available at: http://www.ece.osu.edu/~passino/ECE481LecturesWeb/ECE481Lecture2Codes.pdf
[16] Godfrey, R., “The Compleat Software Engineering Proffesional – Doing the Right Thing as Well as Doing it Right: Five Steps on the Road to an Ethics Curriculum”, in Software Engineering: Education and Practice, 1996, pp. 26-32.
[17] Gotterbarn, D. Miller, K. Rogerson, S., “Computer society and ACM approve software engineering code of ethics”, Computer Volume: 32, Issue: 10, Oct 1999, pp 84-88.
[18] Samaha, Joel. (2002). Criminal Law (7th ed.). Belmont, CA: Wadsworth Group/ Thomson Learning.