Advantages and Trade-Offs of Introducing Ethical Issues in Computing through a Dedicated Course or through Modules in Relevant Content Courses in the Curriculum

William M. Fleischman
Departments of Computing Sciences and Mathematical Sciences
Villanova University
Villanova, Pennsylvania 19085, U. S. A.
william.fleischman@villanova.edu

Daniel T. Joyce
Department of Computing Sciences
Villanova University
Villanova, Pennsylvania 19085, U. S. A.
daniel.joyce@villanova.edu

Abstract. We discuss two alternatives for introducing consideration of ethical questions in the computer science curriculum. These alternatives are 1) a self-contained course on ethical issues in computing, and 2) introduction of modules devoted to ethical questions throughout the curriculum in content courses such as software engineering, databases, data mining, artificial intelligence, and systems. We discuss the advantages and the potential "hidden messages" involved in each of these approaches. By way of illustration, we list some of the pertinent points raised by two important case studies that are appropriate for inclusion in either a self-contained course or a course on software engineering.

Keywords: Computer ethics, Ethical questions in software engineering, Case studies.

1 Introduction

The over-arching goal of consideration of ethical issues in computing has been given paradigmatic expression by Terry Bynum [1]:

> To integrate computing technology and human values in such a way that the technology advances and protects human values, rather than doing damage to them.

It is generally recognized that, in pursuit of this goal, curricula in computer science and computer-related fields should include explicit consideration of ethical issues raised by applications of computer and information technology in building life-

critical, safety-critical, and privacy-critical systems. For undergraduate curricula in the United States, for example, the requirements of the Computing Accreditation Commission of ABET specify that to be approved as accredited any program must present documented measurable outcomes that "enable students to achieve i) an understanding of professional ethical, legal, security, and social issues and responsibilities; and ii) an ability to analyze local and global impact of computing on individuals, organizations, and society" – by the time students are eligible to graduate. [2]

The historical origin of these requirements lies in incidents such as the 1985-86 series of computer-related radiation therapy accidents related to the Therac-25 [3] and the launch in 1987 of the so-called Internet Worm [4]. Since that time, a steady stream of similar stories has provided reinforcement on a regular basis of the need to treat ethical issues in the computer science curriculum [5], [6], [7], [8], [9] and [10].

In the context of the undergraduate computer science curriculum, pursuit of the goal articulated by Terry Bynum often requires appeal to and stimulation of students' imaginations concerning situations they will face five or ten years in the future in the early stages of their professional careers. Without the ability to transcend the relatively protected idea space of their lives as university students, discussion of actual or potential ethical dilemmas may seem artificial and somewhat distant. Thus one important skill that students must develop is the exercise of their powers of imagination and empathic response to help them place themselves in the situation of individuals, often from very different backgrounds than their own, enmeshed in situations involving complex and conflicting power relationships and vulnerabilities. Lacking this ability, students are often give way to the temptation to reduce these problematic situations to simple, one-dimensional self-other oppositions. [11]

2 Some General Observations

Terry Bynum describes three modes of treatment of ethical issues in the computer science curriculum – i) a "stand-alone" course dedicated to ethical issues in computing; ii) the introduction of "case studies in every course" throughout the curriculum; and iii) a "capstone course" in software engineering integrating thorough treatment of ethical issues [1]. In this paper, we will collapse the latter two modes into a single alternative which we will refer to as the "module" approach in which courses in the computer curriculum that have significant technical and scientific content/goals also include modules devoted to ethical issues. Drawing on our own experiences, we will consider the advantages and trade-offs presented by the "stand-alone" course and the approach based on the use of modules.

Since many of the most important and useful examples for discussion of ethical issues come from the area of software engineering, it should be clear how to capitalize on the opportunities afforded adoption of modules in courses directly related to software engineering, including the capstone project-based approach. Still, the examples we cite should also suggest ways to relate some of our insights to other subject areas – for example, artificial intelligence, data mining, and robotics – within the computer science curriculum.

It is worth noting that, in our experience, a case study such as the series of Therac-25 radiation therapy accidents consistently induces astonishment and unhappy surprise among students at the extent of serious harm caused in part by faulty engineering design (involving both hardware and software) of a computer controlled system that they are able to recognize as not unlike systems they may eventually be called upon to implement. We believe strongly that consideration of case studies, carried out with appropriate seriousness and care, has an important role within either of the two approaches that we discuss.

3 Hidden Messages

Before embarking on the discussion of the advantages and trade-offs involved, it seems worthwhile to consider the "hidden messages" conveyed by each of these approaches. It may well be that every course within a given curriculum carries a set of explicit intentions or purposes that are relatively clear and straightforward. On the other hand, the status of the course within the curriculum and the manner in which it is presented often carry hidden messages that can reinforce or subvert the explicit purposes the course is presumed to serve.

What are the messages conveyed by a dedicated, required course in computer ethics? To begin with, this provides a clear signal that those who supervise the curriculum consider the subject important enough to invest precious curricular time to expose students to concepts and case studies that will be treated in depth. Next there are the potential messages associated with the identity of the faculty member who presents the course. If the course is taught by someone from the students' own department, moreover by an individual who takes a serious and informed approach to the material, the tendency will be to reinforce – perhaps very strongly – the importance students attach to the subject. The same effect can be achieved if the instructor is an external individual from the faculty of philosophy or ethics who has nonetheless taken the trouble to cultivate familiarity with the range of questions germane to this area of applied ethics and is able to present at least a few important case studies with an adequate level of understanding of the technical issues relevant to each case. A course taught jointly by a computer scientist and an ethicist would clearly present another favorable situation. On the other hand, a course presented by a disaffected instructor – either a computer scientist or an ethicist who conveyed the sense of having been given an unpleasant or unimportant assignment – would send a strong message of a contrary nature.

The modular approach has the same possibility of sending conflicting hidden messages to computing students. If each course in the curriculum incorporates a well-designed and substantial module, perhaps in the form of a case study, involving a relevant ethical problem, the message to students will be, "This is a subject that is intrinsic to virtually every aspect of the discipline. It engages the attention of all my instructors. I had better be sensitive to similar situations in my professional life." If, on the other hand, the modules are presented superficially or with embarrassment in more than a few instances, the hidden message will be one detrimental to student engagement with ethical problems in their discipline.

4 The Dedicated Course Approach

Here are some of the advantages of a dedicated course in computer ethics:

- Having the time available to lay the groundwork for a common understanding and comparison of a range of ethical theories – deontological or Kantian ethics, utilitarianism, social contract theories, and value ethics.
- Having the time to read and discuss foundational papers in computer ethics.
- The scope of a dedicated course on ethical issues clearly extends well beyond the confines of software engineering or any other single subject area in the curriculum. Thus, the fully dedicated course on computer ethics offers the possibility of treating questions that would not necessarily arise in the context of the capstone experience in software engineering or any other content course. The virtue of this breadth of coverage consists of the possibility of discovering commonalities between situations or case studies that would not be evident when considered in isolation.
- The possibility of treating in depth several case studies which, again, reveal commonalities that underscore the critical importance of various software engineering procedures.
- The possibility of treating in depth several case studies which reveal novel or unexpected aspects of the software engineering process.
- The possibility of combining, in a natural way, ethical, social and legal aspects of the implications of new computing and information technologies. Problems with new technologies rarely come neatly wrapped in a box labeled "Ethical Dilemma: Beware!"
- The possibility of cultivating a large, diverse audience for such a course. Cross-fertilization is a good thing in this context. Individuals from outside the immediate discipline often provide complementary insights to those immediately apparent to students who are absorbed in the details of software engineering practice. The obvious trade-off here is the need to dilute some of the more technical aspects of a particular case study or scenario. (Even this difficulty can be used to advantage by having technically proficient students take the responsibility of explaining the nature and implications of a technical problem to students whose backgrounds are more general, less technical.)

5 The Modular Approach

Here are some of the advantages of incorporating modules devoted to ethical issues in the software engineering curriculum itself:

- Immediate relevance of a case study or question to the course content. For example, in a Systems course, the discussion of dangers and prevention of buffer overflow could be accompanied by consideration of the issues of culpability and responsibility when poor system design leads to serious harm.

- The possibility of motivating a deeper exploration of a technical topic – say, the use of encryption – in connection with a particular example.
- Being assured that the students are sufficiently informed about the technical/managerial/economical issues involved in the issue under consideration.
- Incorporating consideration of ethical issues as a pro-active part of specifying, designing, building, testing, delivering, and maintaining software systems, by building such consideration into the documented approaches taught and used in the specific courses which concentrate on each of these facets of software engineering.
- When consideration of an ethical question can be directly related to a technical project/problem with which the student is actively involved, there is clearly a much higher likelihood that the student will immediately perceive the importance of the topic.

6 Remarks Concerning a Couple of Case Studies

As we have indicated, we consider the treatment of case studies to be an important component of either of the two approaches to incorporating ethical issues into the computer science curriculum. In this section we note – principally in the form of bulleted lists – some of the salient points that can be addressed in connection with two compelling case studies – the series of Therac-25 radiation accidents and the chronic failure of electronic voting technologies in recent U. S. elections. In illustration of some of our earlier assertions, the two sets of bullet points overlap in a substantial number of items. This provides the opportunity to underline the importance and ethical dimensions of questions like the initial stages of design, documentation, testing, and code re-use that students might otherwise consider minor matters.

One particular common item – which we refer to as "ecology of use" – merits further comment. The term "ecology of use" was introduced implicitly by Alvarez and colleagues [12] and explicitly in a recent paper of Fleischman [13]. The concept refers particularly to the situation in which advanced forms of technology, especially life- and safety-critical systems, are placed under the control of technically underprepared personnel. This circumstance places acute emphasis on frequently overlooked elements of engineering and software engineering design, documentation, and testing during the development of such systems. The common link between ecology of use considerations as significant factors contributing to system failure in both the Therac-25 and current electronic voting technologies was discussed by Fleischman [14].

The Therac-25 Accidents [3]

- One can argue that this case study does not actually involve software engineering because when the Therac-25 was developed (the design and programming involved began in the mid-1970s), courses in software engineering were not widely offered by computer science departments.

- But this is clearly one of the "index cases" for incorporation of software engineering in the curriculum and for careful attention to software engineering practice.
- The case of the Therac-25 highlights the importance of good engineering design as a precondition for successful software engineering.
- The Therac-25 accidents point to problems associated with reuse of code.
- The Therac-25 accidents underscore the need for a careful and independently designed regime of testing.
- They also reveal the folly of basing safety on serial elimination of "bugs."
- Considered in a general context, the Therac-25 accidents underscore the importance of documentation in all aspects of development including the need for attention to clear and understandable documentation for non-technical personnel who may have the responsibility for operation of a life- or safety-critical system.
- Again, in the general context, the Therac-25 accidents argue for attention to a broader systems perspective than one that simply focuses on a hardware/software combination. This is sometimes referred to as the "ecology of use."
- Finally, the case study reveals the importance of robust and transparent procedures for government approval and regulation of life- and safety-critical systems.

Electronic Voting System Technology

- Here there are numerous important references including [9], [10], [15], and [16]
- Again, good engineering design must precede good software engineering. Even a system built using the best standards of software engineering can be compromised if system components can easily be "switched out" by someone with physical access to the device.
- Having students see this as a safety-critical technology. (Referring to the Software Engineering Code of Ethics which begins by emphasizing the paramount importance of working to advance the public good.)
- The importance of careful documentation both for purposes of certification and regulation, and for use by election officials and poll workers of varying levels of technological competence.
- Lapses in implementation of state-of-the-art encryption resulting in multiple paths of attack, many of them undetectable, by malicious adversaries.
- Poor or non-existent change control protocols resulting in the possibility of virtually undetectable insertion of malicious code by a malevolent member of the development team.
- Deficient or non-existent testing programs.
- "Ecology of use" considerations relating to the fact that, for poll workers, "every election day is the first (and only) day of work for many, many people." [17]

7 Conclusions

Finally, it is important to say that the approaches described by Terry Bynum are not mutually exclusive. Perhaps the optimal solution for the treatment of ethical questions in computing would combine a dedicated course on the subject with reinforcement (or in some cases anticipation) of relevant issues in a capstone experience or through introduction of short modules in relevant content courses. From the perspective of maintaining currency in a curriculum in which there is always pressure to expand technical content, this may seem to be an infeasibly expensive proposal. One should, however, consider the real costs to society and to the reputation of the individual academic program of producing students who are oblivious to the risks associated with computer and information technology and the potential dangers arising from poor software engineering practices.

Equally, a situation in which the curriculum includes both a dedicated course on computer ethics, appropriately taught, and several instances in which well-designed and meaningful modules are incorporated in content area courses would perhaps represent the ideal form of "hidden message" concerning the centrality of ethical concerns to the discipline.

References

1. Bynum, T.: Computer Ethics in the Computer Science Curriculum, available at http://www.southernct.edu/organizations/rccs/oldsite/resources/teaching/teaching_mono/bynum/bynum_desired_outcome.html, last accessed 15 July, 2011 (2000)
2. ABET Board of Directors: ABET Criteria for Evaluating Computing Programs, available at http://www.abet.org/Linked-Documents-UPDATE/Program-Docs/abet-cac-criteria-2011-2012.pdf, last accessed 15 July, 2011 (2010)
3. Leveson, N., and Turner, C., An Investigation of the Therac-25 Accidents, IEEE Computer, volume 26, no. 7, pp. 18-41 (1993)
4. Eisenberg, T., Gries, D., Hartmanis, J., Holcomb, D., Lynn, M. S., and Santoro, T., The Cornell Commission: On Morris and the Worm, Communications of the ACM, vol. 32, no. 6, pp. 706-709 (1989)
5. Culnan, M. J. and Smith, H. J., Lotus Marketplace: Households…Managing Information Privacy Concerns, Georgetown University School of Business, Case 192-123 (1991)
6. Etzioni, A. Privacy and Safety in Electronic Communications, chapter 4 of The Common Good, Polity Press, Cambridge, MA (2004)
7. Parnas, D. L., van Schouwen, A. J., and Kwan, S. P., Evaluation of Safety Critical Software, Communications of the ACM, vol. 33, no. 6, pp. 636-648 (1990)
8. Singer, P. W., The Ethics of Killer Applications: Why Is It So Hard to Talk about Morality When It Comes to Military Technology, Journal of Military Ethics, vol. 9, no. 4, pp. 299-312 (2010)
9. Feldman, A., Halderman, J., and Felten, E., Security Analysis of the Diebold Accu-Vote-TS Voting Machine, available at http://itpolicy.princeton.edu/voting/ts-paper.pdf, last accessed 15 July, 2011. (2006)
10. Kohno, T., Stubblefield, A., Rubin, A. and Wallace, D., Analysis of an Electronic Voting System, available at http://avirubin.com/vote.pdf, last accessed 20 July 2011 (2003)

11. Fleischman, W., The Role of Imagination in a Course on Ethical Issues in Computer Science, in Proceedings of ETHICOMP 2001: Systems of the Information Society, edited by S. Rogerson, S. Szejko, and T. Ward Bynum, Gdansk, Poland, vol. 1, pp. 171-183 (2001)
12. Alvarez, R., Atkeson, L., and Hall, T., Auditing the Election Ecosystem, working paper # 85 of the Caltech/MIT Voting Technology Project, available at http://vote.caltech.edu/drupal/files/working_paper/wp_85_pdf_4acf9bcad1.pdf, last accessed 15 July, 2011 (2009)
13. Fleischman, W., Electronic Voting Technology, the Software Engineering Code of Ethics, and Conceptions of the Public Good, in The "Backwards, Forwards, and Sideways Changes" of ICT, Proceedings of ETHICOMP 2010, the 11th International Conference, Universitat Rovira i Virgili, Tarragona, Spain, pp. 162-169. (2010)
14. Fleischman, W., Electronic Voting Systems and the Therac-25: What Have We Learned?, in The "Backwards, Forwards, and Sideways Changes" of ICT, Proceedings of ETHICOMP 2010, the 11th International Conference, Universitat Rovira i Virgili, Tarragona, Spain, pp. 170-179 (2010)
15. Theisen, E., E-Voting Failures in the 2006 Mid-Term Elections, available at http://www.votersunite.org/info/E-VotingIn2006Mid-Term.pdf, last accessed 18 July, 2011 (2006)
16. Theisen, E., Vendors are Undermining the Structure of U.S. Elections, available at http://www.votersunite.org/info/ReclaimElections.pdf, last accessed 18 July 2011(2008)
17. Kohno, T., Testimony of Tadayoshi Kohno before the Committee on House Administration of the U.S. House of Representatives Hearing on Electronic Voting System Security, July 7, 2004, available at http://www-cse.ucsd.edu/users/tkohno, last accessed 15 July, 2011 (2004)