Research Ethics and Computer Science: An Unconsummated Marriage [*]

David R. Wright
Computer Science Department
North Carolina State University
Raleigh, NC USA
drwrigh3@ncsu.edu

ABSTRACT

The ethical conduct of research is a cornerstone of modern scientific research. Computer science and the discipline's technological artifacts touch nearly every aspect of modern life, and computer scientists must conduct and report their research in an ethical manner. This paper examines a small selection of potential ethical dilemmas researchers in this discipline face, and discusses how ethical concerns may be addressed in these situations. The paper concludes with an overview of other areas of ethical concern and a look to the future development of a code for ethical computer science research.

Categories and Subject Descriptors

K. COMPUTING MILIEUX [**K.7 THE COMPUTING PROFESSION**]: K.7.4 Professional Ethics

General Terms

Experimentation

Keywords

Codes of ethics, Ethical dilemmas, Research ethics, Responsible conduct of research

1. INTRODUCTION

The social contract between modern scientific research and society rests upon a threefold foundation: the responsible conduct of research; clear and complete recording and reporting of research procedures, results, and analyses; and respect for those that may be affected by that research. Computer science and software engineering, and the technologies the discipline is responsible for, touch nearly every aspect of our world, and researchers in these disciplines bear a great responsibility to the world to conduct and report their research in an ethical manner. Hall and Flynn conducted a survey to investigate the awareness of research ethics among academic computer science and software engineering researchers in the United Kingdom, with discouraging results. Among the results they reported, only 16 of the 44 respondents considered monitoring the ethical considerations of software engineering research to be very important[15]. The other respondents stated they did not have feelings either way (39%), they consider such monitoring not important (18%), or they don't know (7%). Hall and Flynn also reported some striking comments from respondents:

> *"I find this questionnaire very worrying because the idea of having to seek ethical approval threatens academic freedom."*
>
> *"(Seeking ethical approval) has never arisen and I don't know why this is an issue."*
>
> *"No one is responsible for the ethical approval of CS research."*

In this author's experience, including conversations with students and faculty, as well as involvement in recent faculty searches, the primary ethical concerns stated related to plagiarism and cheating. While these are certainly serious issues, they are common to nearly all disciplines. Topics such as measurement techniques, experimental protocols, vaguely stated hypotheses, results, and conclusions, and the identification of at-risk individuals or groups in research are rarely discussed.

The full breadth of these issues is beyond the scope of this paper. Instead, select concerns will be examined to illustrate the potential ethical dilemmas that exist in computer science and software engineering research. Section 2 discusses the selection and use of measurement techniques, and the biases inherent with these metrics, in computer science research. Concerns relating to the recording and reporting of research are covered in Section 3. Section 4 examines the identification of at-risk populations and the responsibility researchers have towards these groups in the conduct of their research.

[*]This paper is based in part upon an essay entitled *Motivation, Design, and Ubiquity: A Discussion of Research Ethics and Computer Science*, available at `http://www.chass.ncsu.edu/langure/modules/documents/Draft6April2006.pdf`

This material is based upon work supported by the National Science Foundation under Grant No. 9818359. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF).

The paper concludes with an overview of other research areas of ethical concern in Section 5.

2. RESPONSIBLE RESEARCH CONDUCT

Empirical science relies on measurements and observation to corroborate or disprove research hypotheses. In computer science, metrics and analysis methods are the primary instruments for measuring software systems. The use of these instruments is complicated by the diverse and dynamic nature of computing, and often by the necessity of integrating the measurement tools into the system under observation. This is further complicated by a lack of consensus on the topic and minimal attention to the quality of measurement results [1]. Computer scientists experimenting with software systems must also be acutely aware that they are observing a fundamentally deterministic, human-created entity that "lives" within a deterministic and human-created execution environment. While the software system and its environment may be highly complex, they remain created constructs and not naturally occurring phenomena, and researchers must take care to avoid and/or account for biases that often exist in these constructs.

In [8], El Emam, et.al. investigated a set of well-known and accepted metrics used to identify the fault-proneness of object-oriented software implementations. By controlling for a variable other researchers had ignored, they demonstrated (both with their own subject software and by repeating prior published experiments) that the metrics did not correlate with the kinds of defects they were supposed to detect. Gustafson found that a benchmarking program that required high-precision input data only checked the results of the floating-point calculations using that input data to four significant digits[14], generating experimental results that could easily be misinterpreted. In [33], Ziles found that a commonly used benchmark to evaluate the performance of health-care systems was seriously flawed: the underlying model was a grossly inaccurate representation of the actual system the benchmark was supposed to simulate.

The Standard Performance Evaluation Corporation (SPEC) CPU2000 suite of benchmarks is an industry-standardized reference for measuring a computer system's processor, memory, and compiler support[27]. Out of 115 papers published in highly respected computer architecture conferences and citing the use of SPEC2000, only 23 used the entire suite[6]. Citron notes that this partial use of a benchmarking suite can be misleading, as readers may not have sufficient information to draw realistic conclusions about the results of the research. Sohi argues that there are sometimes extenuating circumstances that prevent the use of the full suite, and compares this to pharmaceutical researchers who do not test every new drug against all possible diseases[18]. Hennessy counters, in the same panel discussion, that some of this testing does indeed occur, and alternative applications are found for compounds that were not effective for their original purpose. Sohi's comments regarding drug testing procedures fail to recognize that most of the research performed prior to clinical trials is designed to determine the positive and negative side effects of the drugs[30]. Thorough testing, using established tools, can successfully illuminate negative behaviors in software systems that limited benchmarking may overlook.

Another area of bias in software research is the actual implementation of the software itself. Different programming languages have features and semantics that offer more efficient ways to implement certain operations. While these features are generally obvious to the researcher, the optimizations that compilers can perform are often far less visible, and can vary among computer platforms (hardware and operating system). Levy and Clark studied several different benchmark programs, written in different languages and compiled on different platforms. The concluded that the results of these tools (execution time measurements) were not a reliable means of comparing hardware/software systems, the task for which they were designed[22]. Another study found that compilers can create differences in performance using the same program source code as well as compiling and executing on the same platform[13].

The availability of standardized, well-defined means of making measurements is an essential element of science and research. This common ground fosters consensus, collaboration, and rigor[25]. Unfortunately, this situation does not yet generally exist in computer science research. It is critical, however, that the tools that are used are well understood and documented so that researchers can choose the instrument appropriate for the task at hand. Researchers must also take responsibility for the benchmarks they choose to employ, fully disclose the reasons for making those choices, and provide detailed explanations of how the tools are used. It is also essential to accurately and specifically define *what* is measured and *why*.

In [19], Jones is highly critical of measurement and metrics in computer science, noting the measurements reported in publications generally lack the precision necessary to duplicate the work of another researcher and calling the software industry "...and embarrassment when it comes to measurement and metrics." While the focus of this section has been on software benchmarking and measurement tools, the issues and concerns this discussion represents exist across the breadth of computer science research. Performance, quality, reliability, robustness, and security of software systems are all active topics of research that lend themselves to measurement and evaluation. It may be impossible, given the wide range of computing platforms and programming languages, to develop a single, comprehensive set of metrics applicable to all situations. If this is the case, the discipline must at least develop standards for documenting and characterizing metrics, measurement techniques, and benchmarking tools to allow researchers to make informed decisions about the tools they use. Such standards would also provide interested parties outside the discipline to evaluate research results more reliably and accurately. The responsible and repeatable conduct of experimental research is a fundamental part of science.

3. DOCUMENTING AND REPORTING RESEARCH

The ability to duplicate the work of other researchers is perhaps the most fundamental principle and responsibility of science. Repeating an experiment allows a new result to be corroborated or refuted, as well as providing the means to restate and refine the problem under consideration. Duplicating the prior work of other researchers is often also more than simply recreating the earlier experiment: the later researcher should also be looking for new results that extend or clarify the earlier work, or be seeking some case where the existing theory does not apply. It is the continual refinement of hypotheses that builds credibility of both researchers and results.

Rational discourse is a requisite for replicating research. Clear and precise descriptions of experimental setups and protocols, research hypotheses, results anticipated based on theoretical analyses, and complete records of collected data provide the groundwork from which other researchers can attempt to corroborate, refute, and refine existing research. From these attempts to duplicate prior work new, and potentially more interesting, problems become clear. Detailed records can also protect researchers against accusations of fraud or misconduct, just as lapses may indicate their presence.

Given the diversity of research topics within computer science, there cannot be a single standard of judgement for documentation and record keeping. Some experiments, e.g., those studying human reactions or interactions with computing systems, will have requirements closer to those of psychological or cognitive science research, while areas such as experimental algorithmics will not require the same degree of detail[34]. Zobel also points out that, in light of rapidly changing computer technology, it may be nearly impossible to experimentally reproduce results, but that these results should illustrate the same underlying phenomena.

Defects in software are closer to the norm rather than the exception, and programming errors can result in false positives as well as false negatives in research. Skepticism and caution must be equally applied to both the justification of positive results as well as the mitigation of negative outcomes. Researchers have only one tool to justify the correctness of their experimental approaches, software designs and implementations, experimental environments, and collected data — their own detailed records and documentation of their work. Zobel points out that the general feeling that "the records are in the software" and associated documentation is insufficient and often irresponsible: the software is the *result* of a collection of decisions, experiences, and reevaluations[34]. He goes on to recommend that computer science researchers adopt the mandatory practices of other disciplines, procedures such as daily research journal entries, maintaining *all* versions of experimental software, and detailed logs of all collected data. The nature of software and the virtual environment in which it exists does not permit short-cutting the essential tasks of researchers. Rather, this nature and environment require as much, if not more care, discipline, and effort than may be common in other fields.

Given the ubiquity of computing in our world, it is imperative that computer scientists embrace and interact with other disciplines[10, 21]. Hartmanis asserts that computer science must "increase its contact and intellectual interchange with other disciplines," and avoid focusing on distinctions between basic and applied research and development, and embrace the transfer of knowledge between academic, industrial and social researchers[16]. It is the responsibility of computer science researchers to build upon the work of others, while creating new knowledge and understanding to extend and strengthen the discipline. For a young discipline or subarea, it is often critical to look outside the immediate bounds of that research area to similar fields in other areas of investigation. Unfortunately, that outward-looking attitude is the exception rather than the rule in computer science and software engineering: 89% and 98% (respectively) of literature references were within the field of study according to one survey of publications[11]. Not only does this introversion place limits on possible insights and innovations, it reflects a separation from the larger community of science and society.

This interaction with other disciplines necessitates acceptance of the same level of detail and documentation expected in these other research areas. One key to this is the development of clear and specific research hypotheses as opposed to generalizations and abstractions. Generalizations may be easy to corroborate, but they ultimately tell us little about the world or the phenomena they are supposed to explain. Popper notes that the *empirical content* of a statement increases with its degree of falsifiability, the number of ways that the statement can be proven untrue, and that the more specific a theory is, the more it forbids, and the more it tells us about the world[23, p.119]. Studying publications in computer science and software engineering, Glass, et.al., found that the majority presented abstract and formulative research findings[11, 12]. In his 1993 Turing Award lecture, Hartmanis uses this emphasis on the abstract *how* to justify computer science's distinction as a new kind of science[17].

In contrast to this perspective, Stewart responds by criticizing the vague and often unstated problems and hypotheses that characterize some areas of research in the discipline[28]. Tichy, et. al. are less kind, calling the lack of experimental evaluation in computer science "unacceptable, even alarming," and echoing Jones[19] characterization of the software industry as "amateurish craft" and an "embarrassment." They go on to discount explanations regarding the youth of the discipline, the difficulty of experimentation, and the fear of negative career impact, giving the unstated implication that the root cause may be simple laziness, or a desire to not figuratively rock the boat. The risk of damage to the discipline is great, and because of the intimacy and pervasiveness of computing today, the risk to society as a whole is even greater.

The dilemma computer science and software engineering faces is one of respectability and leadership. To achieve the respect from established scientific disciplines, computer scientists must adopt and adhere to comparable standards of recording and reporting their research, particularly when that research has close parallels with other disciplines. In some areas within the umbrella of computer science, this is already the case. Human-computer interaction is a notable example — this area has adopted the standards of cognitive science and psychology principally because of the involvement of psychologists and other cognitive scientists in collaborative research. Clear and comprehensive documentation and publication is also the path to leadership because it demonstrates the maturity of the discipline and the awareness of researchers of their place in the larger scientific community. In most disciplines research drives technological advance but the rapid pace of change in the computing industry often results in research into problems created by the commercial proliferation of new technologies. All too often the marketplace is the laboratory (and software companies the researchers) where new problems are identified and solutions tested. The pervasiveness of computing in the modern world seeks leadership from computer science, and in practice, that leadership has been weak. As computer scientists, we have the responsibility to be leaders and not followers. Paraphrasing Isaac Asimov's Zeroth Law of Robotics[4]: "A computer scientist may not injure humanity, or, through inaction, allow humanity to come to harm."

4. HUMAN PARTICIPANTS IN COMPUTER SCIENCE RESEARCH

The topic of human subjects and participants in scientific research is a key element of research ethics. In computer science, this topic is probably most closely associated with the study of human-computer interaction, with the evaluation of computer science educational methods and software development techniques following closely behind. Most research institutions have some form of a review board that evaluates research proposals involving human subjects. Key requirements for approval include obtaining uncoerced informed consent to participate from research subjects, and providing the option to withdraw from the study at any time and for any reason without penalty. However, there are areas of computer science research that involve humans in more subtle ways but that can still put individuals at risk to harm. In this section, we examine some of those cases and identify the risks that may be involved.

The first case involves research using *open source software* (OSS). OSS is attractive to researchers because they have access to the source code of large, complex software systems, access that is generally not possible with commercial software systems. In [9], El Emam proposes a research scenario where an experimental metric is applied to an OSS system to evaluate the quality of the source code. Noting that OSS is publicly and freely available, he questions the need to obtain informed consent from the developers, but also raises the issue (of particular concern to ethicists) that the contributors to the project did not intend for their work to be used as subject material in software metrics research. However, the task of getting consent from everyone involved in an OSS project is daunting: there may be hundreds or even thousands of contributors, and a researcher cannot know in advance which particular individual's contributions are going to be the subject of analysis.

El-Emam also raises questions regarding minimization of harm and confidentiality. Version control systems used in OSS projects tag contributions with information identifying the developer responsible for each piece of code integrated into the system. If the research results in the publication of code segments, the source code could be traced back to the individual who wrote it, and possibly cause that person professional or personal harm (particularly if the source code was presented in a negative context). Vinson and Singer respond by noting that eliminating personal identifiers from the reported data in order to maintain confidentiality reduces the need for informed consent, but does not necessarily eliminate it[32]. Removing all traces of personal identification is a daunting task, however, since it may be necessary to conceal not only the identities of the developers, but also the identity of the OSS project itself. Including an exemplar piece of source code from an identifiable project in a publication would allow someone to search that project's code repository, and from that identify the developer that contributed it. Not identifying the source of the subject software is also problematic for other researchers because it hinders attempts to duplicate the initial research. Vinson and Singer conclude that research with OSS is "fraught with ethical issues," and encourage researchers to be proactive in the development of ethical guidelines for this type of research.

Other cases involve the recruitment of students or employees as research subjects. Students may be inadvertently coerced into volunteering as research subjects by offering course credit for research-centered classes or extra credit for a "research assignment" as part of a regular class. Storey, et.al., raise several ethical issues in addition to this covert inducement: the fairness between students who dropped out and those who did not, the involvement of the instructor and teaching assistant as experimenters in the research, the stress of learning and switching between new and unfamiliar tools, as well as extraordinary time demands on the teaching staff due to questions about and defects in the systems under evaluation[29]. Davis questions why the institution's ethics review committee did not identify the coercion in the experiment proposed by Storey[7].

Similarly, employees can also be subject to psychological, social and economic harm when research is conducted in the workplace[24]. In [26], Singer and Vinson identify several other ethical problems with workplace research. Employers often desire to maintain control over their employees and their work, and this control may interfere with the research protocols. Employers may also be interested in the results of the research, particularly if they are funding it, and may want access to the detailed data collected. This data could contain personally identifiable information which could then put an employee at risk for harm with respect to advancement within the company. There is also the potential conflict of interest between the good of the research, the good of the employee/subject, and the good of the company.

Less obvious cases of potential risk to individuals can be found in other research areas within computer science:

- Graphics and Virtual Reality: Alternative representations of non-graphic information, creation of (nearly) complete sensory environments that may or may not exist in reality are very human-subject intensive, as the representations created and manipulated are intended for human perception.

- Knowledge Discovery and Data Mining (KDDM): Large and diverse data sets are examined and manipulated for the purpose of extracting "hidden" knowledge and patterns not otherwise observable, often using "agents" as faster and tireless human surrogates. The potential exists for the invasion of personal privacy. There is also the possibility of misleading results (agents or programs are looking for particular kinds of patterns that a researcher expects to find in a particular data set, etc.) that could cause harm to individuals or groups.

- Networking: There is also the risk of potential invasion of privacy when monitoring live networks. An ethicist might also ask if the users of a computer network that is a subject of study are themselves subjects in the study? After all, it is their usage that creates the patterns of traffic that are of interest to a researcher.

- Software Engineering: A significant amount of research in this area is very human-involved, since it is humans that design, implement, use, and maintain software systems. Research often involves observing individuals, groups, and entire organizations, with the associated risks of harm.

- Computer Science Education: Trying to keep up with the rapid pace of technological change has stressed CSE greatly. Many courses (particularly introductory-level) are somewhat experimental as educators try to find new ways to educate ethnically, gender, age, and socio-economically diverse students. Often this research involves investigators trained in educational research, but this is not (by far) a universal phenomenon.

- Cryptography: Cryptographic strength needs to be tested and evaluated, but what happens if a researcher finds a more efficient means to crack a cipher? Publication could have serious and immediate negative effects for large numbers of individuals as well as for many organizations, but also promotes the greater good by identifying weaknesses and strategies for their correction.

5. THE UNCONSUMMATED MARRIAGE

As the title of this paper states, the union between research ethics and computer science has not been consummated. While the discipline has established codes of ethics and professional conduct (see [2] and [5]), these codes are oriented more towards professional practitioners than researchers in the field. They fail to address issues that might be critical in a research context but would not occur in normal professional practice. An example of this is the use of human subjects in research: researchers must take extensive precautions to mitigate the potential harm to research subjects, while practitioners are explicitly prohibited from using "computing technology in ways that result in harm to any of the following: users, the general public, employees, employers"[5].

The analogy exists in the practice and research of medicine: physicians are charged with holding the responsibility to their patients and their well-being paramount[3], yet pharmaceutical researchers give *healthy* volunteers new drugs in Phase 1 clinical trials to determine side effects, dosing, and drug metabolism[31], a procedure which can have significant risk for the subject. Similarly, we would not permit a civil engineer to design a bridge that had a high probability of failure, yet researchers will do just this to understand *why* these structures fail. Any scientific research has some degree of risk by its very nature — investigating the unknown. Researchers are obligated to mitigate the risks as much as possible, and most importantly, to inform those who may be affected by their research of the possible risks prior to subjecting them to those risks.

Researchers also have the obligation to thoroughly document and disseminate their work to betterment of the world we live in. This implies that research drives technology, a situation that is somewhat reversed in computer science. As a discipline, we expend significant effort trying to understand the implications of technology already in use. In some respects, this is understandable, since computers and the software systems that control them are human artifacts, and their attempted application to real-world problems is constrained only by the imagination of those who develop these systems. That imagination and its application is entrepreneurial — technology developers are profit-seekers, and many more fail than succeed. This does not relieve computer science researchers of their leadership responsibility. Trained researchers have the obligation to evaluate new technologies for their safety and efficacy, to establish standards and criteria to guide evaluation, and develop the metrics and methods to allow others to perform those evaluations. Computer scientists have a greater opportunity to publish their results to the world than in any other discipline because of the multitudes of journals, conferences, and symposia available to them. They also have the obligation to critically examine the work of others, and to duplicate prior work to corroborate or disprove it and to find new insights.

Computer science also has its own controversial research areas, and its own "forbidden knowledge." The goal of artificial intelligence (AI) research has been to develop computer systems that are able to act autonomously, to learn from their past behavior, and to behave intelligently. Ethical implications researchers must consider include:

- How do we define *intelligence*, *autonomy*, and *sentience*?

- Would a sentient computer have the same legal rights as a human? Would the answer be different if the computer were biological rather than electronic?

- What rights would a human being have with regard to knowing when they were interacting with an artificial intelligence vs. and real human being?

- Is it murder to "turn off" an intelligent, autonomous entity?

Research into malicious software is generally limited to textbook analysis of threats, the behavior of malicious code (e.g., viruses, worms, etc.), and the flaws that these programs exploit. Students are not taught how these programs are actually written, and rarely get the opportunity to examine the inner workings of them. These are dangerous entities, with serious moral and legal consequences if they escape into the world. Research publications in this area, based on a quick review of the titles and abstracts of the first 50 of 200 papers returned by a ACM Digial Library search on the terms "malicious software" "worm" and "virus," seem to support this perspective. These publications focus on securing computers against attacks, the ethical issues surrounding the public release and dissemination of malicious code, and modeling the spread of malicious programs through various distribution channels. None of these papers claimed to have examined the internal structure of the infectious programs — all analysis was based on external observations. Ledin recommends teaching the internal structure and workings of malicious programs for the same reasons scientists study deadly diseases and common infectious agents in everyday life: understanding the disease is fundamental to developing a cure, not just a treatment[20].

Researchers in computer science are somewhat isolated from the world by their computer screens and keyboards, but the ubiquity of computing in the world today demands that we assume the responsibility to society that comes with that position. The consummation of this marriage will not occur in a single, blissful moment, but as a process that will rightly take several years to come to fruition, and even then will not be a single, static event but a dynamic process that evolves with the growth of our knowledge and technological experience. This paper represents a step in that direction: an attempt to build awareness and instigate dialog within the discipline towards the development of ethical standards for computer science research. The path will not be easy, but the outcome will be invaluable, not only to this discipline, but to the world in general.

6. REFERENCES

[1] A. Abran and A. Sellami. Measurement and metrology requirements for empirical studies in software engineering. In *Software Technology and Engineering Practice, 2002. STEP 2002. Proceedings. 10th International Workshop on*, pages 185–192, Washington, DC, 2002. IEEE Computer Society.

[2] ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices. Software engineering code of ethics and professional practice. *SIGSOFT Softw. Eng. Notes*, 24(1):10–14, Jan 1999.

[3] American Medical Association (AMA). Principles of medical ethics. Web document, June 2001. `http://www.ama-assn.org/ama/pub/category/2512.html`, accessed 6/15/06.

[4] I. Asimov. *Robots and Empire*. Ballantine Books, New York, 1985.

[5] Association for Computing Machinery. ACM code of ethics and professional conduct. http://www.acm.org/constitution/code.html, Oct 1992. accessed 04/03/06.

[6] D. Citron. MisSPECulation: partial and misleading use of SPEC CPU2000 in computer architecture conferences. In *ISCA '03: Proceedings of the 30th annual international symposium on Computer architecture*, pages 52–61, New York, NY, USA, 2003. ACM Press.

[7] M. Davis. When is a volunteer not a volunteer? *Empirical Software Engineering*, 6:349–352, 2001.

[8] K. El-Emam, S. Benlarbi, N. Goel, and S. N. Rai. The confounding effect of class size on the validity of object-oriented metrics. *IEEE Transactions on Software Engineering*, 27(7):630–650, July 2001.

[9] K. El-Eman. Ethics and open source. *Empirical Software Engineering*, 6:291 – 292, 2001.

[10] P. A. Freeman. Effective computer science. *ACM Comput. Surv.*, 27(1):27–29, 1995.

[11] R. L. Glass, V. Ramesh, and I. Vessey. An analysis of research in computing disciplines. *Commun. ACM*, 47(6):89–94, 2004.

[12] R. L. Glass, I. Vessey, and V. Ramesh. Research in software engineering: an analysis of the literature. *Information and Software Technology*, 44(8):491–506, Jun 2002.

[13] S. T. Gurumani and A. Milenkovic. Execution characteristics of SPEC CPU2000 benchmarks: Intel C++ vs. Microsoft VC++. In *ACM-SE 42: Proceedings of the 42nd annual Southeast regional conference*, pages 261–266, New York, NY, USA, 2004. ACM Press.

[14] J. Gustafson. Purpose-based benchmarks. *Int. J. High Perform. Comput. Appl.*, 18(4):475–487, 2004.

[15] T. Hall and V. Flynn. Ethical issues in software engineering research: A survey of current practice. *Empirical Software Engineering*, 6:305 – 317, 2001.

[16] J. Hartmanis. Computing the future: committee to assess the scope and direction of computer science and technology for the national research council. *Commun. ACM*, 35(11):30–40, Nov 1992.

[17] J. Hartmanis. Turing award lecture on computational complexity and the nature of computer science. *Commun. ACM*, 37(10):37–43, 1994.

[18] J. Hennessy, D. Citron, D. Patterson, and G. Sohi. The use and abuse of SPEC: An ISCA panel. *Micro, IEEE*, 23(4):73–77, Jul-Aug 2003.

[19] C. Jones. Software metrics: good, bad and missing. *Computer*, 27(9):98–100, Sep 1994.

[20] G. L. Jr. Not teaching viruses and worms is harmful. *Commun. ACM*, 48(1):144, 2005.

[21] N. G. Leveson. Software engineering: stretching the limits of complexity. *Commun. ACM*, 40(2):129–131, 1997.

[22] H. M. Levy and D. W. Clark. On the use of benchmarks for measuring system performance. *SIGARCH Comput. Archit. News*, 10(6):5–8, 1982.

[23] K. Popper. *The Logic of Scientific Discovery*. Basic Books, New York, 1959.

[24] J. E. Sieber. Protecting research subjects, employees and researchers: Implications for software engineering. *Empirical Software Engineering*, 6:329–341, 2001.

[25] S. E. Sim, S. Easterbrook, and R. C. Holt. Using benchmarking to advance research: a challenge to software engineering. In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, pages 74–83, Washington, DC, USA, 2003. IEEE Computer Society.

[26] J. Singer and N. Vinson. Ethical issues in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 28(12):1171 – 1180, Dec 2002.

[27] Standard Performance Evaluation Corporation. CPU2000. http://www.spec.org/osg/cpu2000/, 2000. accessed 04/03/2006.

[28] N. F. Stewart. Science and computer science. *ACM Comput. Surv.*, 27(1):39–41, 1995.

[29] M. Storey, B. Phillips, and M. Maczewski. Is it ethical to evaluate web-based learning tools using students? *Empirical Software Engineering*, 6:343–348, 2001.

[30] U.S. Food and Drug Administration. The beginnings: Laboratory and animal studies. `http://www.fda.gov/fdac/special/testtubetopatient/studies.html`, Jan 2006. accessed 04/03/2006.

[31] U.S. Food and Drug Adminstration. Inside clinical trials: Testing medical products in people. `http://www.fda.gov/fdac/special/testtubetopatient/trials.html`, Jan 2006. accessed 04/03/2006.

[32] N. Vinson and J. Singer. Getting to the source of ethical issues. *Empirical Software Engineering*, 6:293 – 297, 2001.

[33] C. B. Zilles. Benchmark health considered harmful. *SIGARCH Comput. Archit. News*, 29(3):4–5, 2001.

[34] J. Zobel. Reliable research: Towards experimental standards for computer science. In J. Edwards, editor, *Proceedings of the Australasian Computer Science Conference*, pages 217–229, Perth, Western Australia, Feb 1998. Springer-Verlag.