New Orleans JAZZED about Engineering Education

Paper ID #15871

On the Integration of Ethical, Legal, and Societal Issues into a Computer Science Senior Design Capstone Program

Dr. Shawn Bowers, Gonzaga University

Dr. Bowers is the Chair and an Associate Professor of Computer Science within the School of Engineering and Applied Science at Gonzaga University. He graduated with a PhD in Computer Science from the OGI School of Science and Engineering at OHSU. He was a postdoctoral researcher at the San Diego Supercomputer Center at UCSD and an Associate Project Scientist at the UC Davis Genome Center prior to joining the faculty at Gonzaga. His research interests are in the area of data management, conceptual modeling, and workflow systems.

Dr. Ellen M. Maccarone, Gonzaga University
Dr. George D. Ricco, Gonzaga University

George D. Ricco is the KEEN Program Coordinator at Gonzaga University in the School of Engineering and Applied Science. He completed his doctorate in engineering education from Purdue University's School of Engineering Education. Previously, he received an M.S. in earth and planetary sciences studying geospatial imaging, and an M.S. in physics studying high-pressure, high-temperature FT-IR spectroscopy in heavy water, both from the University of California, Santa Cruz. He holds a B.S.E. in engineering physics with a concentration in electrical engineering from Case Western Reserve University. His academic interests include longitudinal analysis, visualization, semantics, team formation, gender issues, existential phenomenology, and lagomorph physiology.

On the Integration of Ethical, Legal, and Societal Issues into a Computer Science Senior Capstone Program

1. Introduction

Topics in professional ethics play an important role in ABET accreditation of computer science programs, where ethical issues are mentioned within three of the eleven ABET computer-science student outcomes. To help address these outcomes and to further develop topics in professional ethics within our program, we have developed a set of modules covering ethical, legal, and societal issues in computer science that we have integrated into our year-long senior capstone program. According to other studies[1], a lack of technical knowledge and sophistication are often seen as barriers to student engagement in ethics courses taught in lower-level courses. Thus, one of our reasons for covering ethical considerations at the senior level is to help make ethics more concrete and tangible for students by leveraging their experience and maturity in software development (gained through coursework, internships, and in thinking about their own careers). Further, the approach of integrating ethics modules into a senior capstone program is relatively unique compared with the teaching of ethics in other computer-science programs[2], where fewer than 6% integrate their ethics content into senior capstone projects. However, by integrating ethical considerations into the senior capstone, students can use their own senior projects as (concrete) cases for exploring ethical issues. Our experience suggests that the students' own projects form a well-understood and personal context that allows them to quickly understand and apply ethical, legal, and societal considerations in computing.

In this paper, we describe the modules we have developed, how we have used these modules within the senior capstone program over a two-year period, and future changes based on our experiences to this approach. Although new for our program (i.e., only used over the past two years), we feel there are significant advantages of introducing ethics at the senior level and within the context of a senior capstone program, and intend to continue the approach with only small modifications. Also, for a small to mid-sized computer-science program such as ours (graduating between 25 and 35 students per year), integrating ethics within the capstone program does not require the use of additional teaching resources (e.g., that would be needed to offer a stand-alone course dedicated to ethics), while still meeting ABET requirements related to covering ethical, legal, and societal issues in computing.

The rest of this paper is organized as follows. Section 2 provides a brief overview of approaches for teaching domain-specific ethics within computer science programs. Section 3 describes our senior-design capstone program, which forms the foundation for our approach.

Section 4 describes the history and impetus for our approach to offering ethics within the capstone program. We also describe preliminary data on students understanding of professional ethics prior to working through the modules. Section 6 provides an overview of the modules we have developed and are currently using. Section 7 summarizes our work in progress as well as future changes to our approach.

2. The teaching of professional ethics in computer-science programs

According to Quinn[2], many computer science departments have their students take ethics through courses taught in other departments (e.g., philosophy), only to later develop their own computer ethics courses for accreditation purposes. The computer-science specific ethics courses are often developed after having difficulty demonstrating to accreditors that their students were exposed to ethical considerations. Program evaluators often want to see graded student work related to the domain-specific ethics outcomes, which if taught outside the program can be difficult to obtain (or even require). The current ABET Computing Accreditation Committee student outcomes related to ethics for computer science require students to attain:

- An understanding of professional, ethical, legal, security and social issues and responsibilities;
- An ability to analyze the local and global impact of computing on individuals, organizations, and society; and
- A recognition of the need for and an ability to engage in continuing professional development.

The last outcome, while not specifically tied to ethics, plays an important role in the ACM/IEEE Software Engineering Code of Ethics[6], which is the primary code of ethics used within computer science. Specifically, the code of ethics is divided into the following eight sections:

1. PUBLIC - Software engineers shall act consistently with the public interest.
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.
3. PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.
5. MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.

8. 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.

Each section (i.e., principle) is further elaborated into a set of clauses (80 in total) that illustrate obligations of a professional software engineer. Lifelong learning being one such area of obligation.

Godweber et al.[1] argue that while many programs cover topics relating to computer history, codes of ethics, and intellectual property, many do not cover issues of societal impact having to do with cultural issues, accessibility issues, computing and public policy, the impact of free and open-source software, and so on. The authors argue that one approach for introducing such topics into a curriculum is through capstone projects, e.g., that develop products to serve the public good. A number of authors[1, 2, 7, 8] have developed "best practices" for teaching computer science ethics. Best practices typically involve the use of case studies, hands-on exercises, role-playing, discussion sessions, written assignments, and project work.

Quinn[2] surveyed 50 ABET accredited computer-science departments (out of 200 programs at the time) concerning the coverage of ethics within their curriculum. Approximately half of the programs taught a stand-alone ethics course, where two-thirds of these were designed to be taken by students in their first-three years (as either freshmen, sophomores, or juniors) and the other third taken by seniors. The remaining half of programs either integrated ethics within existing courses or had students take ethics courses taught by other departments (e.g., Philosophy). We note that until recently, our program fell into the latter, where our students received ethics-related content through a junior-level philosophy course (i.e., taught by the Philosophy Department at our university). Further, only 10% of all programs integrated ethics into their software engineering courses and only 6% integrated ethics into a senior capstone program. A similar study was conducted by Barroso and Melara[3] in which approximately 54% taught computer ethics in a stand-alone course and 20% in the context of another subject, with a similar distribution of courses offered across students' four years of study. Spradling et al.[4] reported that smaller programs tended to integrate ethical issues across courses in their program (as opposed to having stand-alone computer ethics courses), which is the recommended approach for small programs with few resources according to Martin[5]. Thus, according to the data available, a majority of programs teach professional ethics "in house" within their program (either as standalone courses or integrated within existing courses). A much smaller percentage of programs integrate ethics into software-engineering related courses, including a senior capstone course (only 6% of programs), which is the focus of our integrative approach.

The remainder of this paper reports on our experiences integrating ethical content within our senior capstone program to help satisfy ABET requirements and to provide, in general, a richer design and capstone experience for our students. As mentioned above, there are a number

of potential benefits to such an integration, and we touch on a number of the advantages we have found using this approach below.

3. An overview of the senior design capstone program

Our computer-science senior-design capstone program consists of a two-semester course in which teams of 4-5 senior-level students work on a software development project proposed by an external sponsor (and in some cases, student-initiated projects with external project clients). Every project is advised by a faculty member and evaluated by both the faculty and members of an advisory board made up of local computing professionals. As part of the program, students are assessed on a number of factors including the quality of the software they produce and whether the software creates value for their customers and end users. All teams are expected to meet frequently with their sponsors, produce multiple prototypes that they receive feedback on throughout the year, and by the end of the year, have a fully functional, high-quality software product that has been tested and accepted by the sponsor. Examples of projects we have had over the past three years include:

- Software for wearable hardware designed to help people with Parkinson's Disease speak at a consistent volume as part of their speech therapy (where the software automatically detects when patients are speaking softly and tracks patient's speech data over time)
- A system for automating search-engine optimization for domain ranking improvement
- A web app for supporting student peer evaluations of work within academic courses
- A mobile app to reduce medical waste in hospitals by providing a mechanism to resale unused medical devices among hospitals
- A mobile app for a ski resort that allows users to track their vertical feet skied and other skiing statistics, a leaderboard that ranks skiers on their statistics, a social share feature (via Facebook) to post and comment on accomplishments, and a push-notification service
- A novel distributed hash-table implementation for secure file sharing (designed to support distributed certificate authorities)
- A web app that monitors energy and water usage across campus with a competition feature for reducing energy consumption (e.g., among student dorms)
- A mobile app that gamifies the tracking of personal attendance at fitness studios to calculate individual fitness profiles and promote strength, flexibility, and cardio balance
- A desktop application for controlling an electric winch designed to launch glider planes
- A free-and-for sale mobile application for students to increase campus sustainability and reduce waste as students move in and out of dorms
- A mobile and wearable app for tracking sleep and sleep influences to help people with sleeping disorders and issues
- A financial services web app for managing government tax liabilities within the telecommunication sectors

- A wayfinding mobile app for a large regional hospital
- A GPS tracking app for runners (integrated with Facebook) so that during races so that their friends can track their progress
- An irrigation monitoring and control system for small-to-medium sized orchards

While only a sample of the projects over the past three years, each project offers unique, real-world, and sometimes complex opportunities to study ethical considerations (such as privacy, security, potential user harms, legal issues, accessibility issues, and issues surrounding intellectual property) as well as potential societal impacts and the inherent values of the surrounding technologies.

The first semester of the program is organized as a traditional course in which students learn software engineering techniques that they apply to their projects, including requirements engineering, risk assessment, estimation and scheduling, project management, and design and development approaches for large-scale software projects. Students are expected to create project plans, give presentations, and develop working prototypes of their software by the end of the semester. Traditionally, the second semester has consisted of fewer software engineering topics, and a greater emphasis on using class time to allow students to work on their project through various in-class exercises. These exercises cover a range of topics designed to help students complete their projects including project management and planning (e.g., through pre-mortems, refining schedules, etc.), software design and design reviews, code reviews, and system and usability testing. The ethics modules we have developed fit into this model by incorporating a number of additional in-class exercises.

As seniors, the students have had a wide range of computer science courses and typically take additional computer science electives during their senior year. They have already finished their core computer science courses in programming, data structures and algorithms, programming languages, operating systems, and computer architecture and assembly language programming. They have also typically taken three or more upper-division elective courses in computer science. The majority of the students taking the capstone course sequence have also done at least one summer internship in some form of software development (although this is not a requirement).

In addition, all students are required to take a 300-level philosophy course in ethics during their junior year as part of the university's core curriculum. This ethics course is taught through the Philosophy Department and covers general ethical theory and applications to various moral problems (the specific problems depend on the section of the course taken). Thus, students entering the senior capstone program have a broad background in computer science and have been exposed to foundational issues in general (philosophical) ethics. In contrast, the ethics modules developed for the capstone program provide specific topics in professional ethics

related to computing, including topics in legal issues, societal issues, and professional development.

4. Motivation, development, and preliminary results of the ethics modules

Our program was only recently accredited by ABET, which motivated in large part the need for our department to formalize and expand the ethics curriculum of our program (which others have identified as one of the main reasons programs begin teaching their own ethics content[1, 2]). Historically in our program, professional ethics was taught within a dedicated software engineering course. This software engineering course was then removed from the curriculum and integrated into the first-semester of the senior-design capstone course four years ago to make room for additional computer science electives and required courses (in part, for reasons associated with ABET accreditation). This left a need to cover ethics within the curriculum, and it was decided within the department to pursue the teaching of computer science ethics within the second semester of the capstone program.

We began by working with two professors in the Philosophy Department to expand the topics covered with respect to ethics into a broader treatment of professional ethics and the societal impacts of technology (and specifically, software). The two professors are experts in the areas of applied ethics and the philosophy of technology, respectively. This work led to two one-and-a-half week modules that each philosophy professor taught within the second semester of the capstone course (in the Spring of 2014). These "mini-courses" included in-class exercises as well as essay assignments on professional ethics and societal impacts. Much of the material was distilled from existing courses taught by the two professors as upper-division philosophy electives.

We further developed these modules into a more comprehensive set of lecture notes, in-class exercises, case studies, assignments, and evaluation rubrics the following year. Our goal was to expand the number of topics to better cover the ABET student outcomes (including legal and security issues, and professional development), to expand the assessment approach used to better align with our ABET assessment procedures, and to make the modules reusable for other faculty in the department. This new set of modules was initially taught in the Spring of 2015 and is being taught again in Spring 2016. The current modules incorporate software-engineering specific material developed by Vallor and Narayanan[8], material on legal and ethical issues (similar to material from Baase[10]), and content from various other sources (including the ACM/IEEE Software Engineering Code of Ethics). The major emphasis of our approach has been to ground ethical, societal, and legal issues within the context of student capstone projects. Specifically, we want to help students gain a deeper, more nuanced perspective on ethical, legal,

and societal issues in computing through the software engineering experiences provided by their senior design projects.

4.1 Pre-survey results of students prior to taking the ethics modules

We developed and implemented a pre-survey prior to our most recent offering of the ethics modules to gain a better understanding of students understanding of and personal views on ethics in computer science. The survey was given to the students during the second semester of the capstone sequence, one week prior to the start of the ethics modules. The survey consists of questions divided into four categories: general questions on ethics; professional ethics; legal and security issues; and specific questions targeting the ACM/IEEE code of ethics. All 31 students taking the senior capstone course completed the survey. We briefly summarize the results of the pre-survey below, which in general suggest that students do not have similar or strong beliefs concerning computer science ethics as a group.

Nearly all of the students had taken the 300-level required ethics course, and three of the respondents had taken an advanced ethics course at the university. However, the students who took the advanced ethics course did not answer differently (either statistically or from visual observation) from those who had not taken the advanced ethics course. When asked to rank which disciplines have the "least" to "most" need for ethics, there was a wide variety of answers. For instance, the same numbers of students (4) ranked computer science, computer engineering, biomedical engineering, and mechanical engineering as having the least need for ethics; 14 of the students ranked biomedical engineering as having the most need for ethics; and only 1 ranked civil engineering as having the most need for ethics. Computer science's distribution curve, i.e., the distribution of ranking for computer science relative to the other disciplines, put it in the middle of all of the other fields, and was more heavily weighted towards "less" than electrical, mechanical, biomedical, transmission and distribution (power), and engineering management. In contrast, over 80% of respondents mostly disagreed, disagreed, or strongly disagreed with the statement "Compared to other fields, software developers rarely need to worry about the unintended consequences of the products they create." The same distribution emerged with the question "It isn't as important to learn new skills and techniques in software engineering as it is in other professions." Once again, over 80% of students answered in the "mostly disagree" column or higher. Analogously for retraining of new skills, the inverse was recorded for "I believe that learning and using new software engineering skills will be important for advancing my career," with nearly all student answering in the affirmative (and exactly 50% with "strongly agree"). Thus, students hadn't made the connection between professional development and professional ethics.

Students answered more conservatively on questions such as “I have a responsibility to educate the public about computer science” (taken directly from the ACM/IEEE code of ethics), with a centered distribution around mostly agree/disagree, as well as more ambiguous legal questions such as “I feel that liability laws are clear and protect consumers from defective and/or insecure software,” and “I should be able to copy software for personal use without repercussions.” Students did feel strongly about the need for “fair use” in software legal issues, with about 80% in the affirmative on the question, “I should be able to copy software for personal use without repercussions,” as well as “I should have the right to use and extend software written by others for noncommercial, personal use.” This, of course, runs contrary to the strong legal protections (through end-user licensing) granted to the majority of commercial software products written today that students use frequently within the university for their work.

Contrary to the students’ beliefs that reverse engineering and non-commercial use should be allowed under the law, they also felt that source code in general should be protected “under the same laws as inventions of physical objects.” A similar distribution (over 75%) answered that code should be protected as those who believed that they should be able to extend software written by others for non-commercial use. This dichotomy is interesting and will be further explored as our work progresses. Overall, these and the results of similar areas of questioning on the pre-survey suggest that students have not formed strong opinions or have thought in depth about computer science ethics.

4.2 Post-survey results of students after taking the ethics modules

When re-surveyed after the completion of the ethics modules (approximately six weeks after the initial survey), a number of interesting results were observed. First, the distributions of question 9, “Which engineering disciplines have the least (top) to most (bottom) need for ethics?”, were significantly changed. Student opinion shifted from least to most ranking in a way that seemed to indicate they were developing a greater appreciation for subtle issues in ethical thinking. For example, the more “theoretical” fields that were rated lower before, were ranking more important in the follow-up survey. Fields such as electrical engineering, which garnered most of its votes in the 2-4s, now were centered around 4. Also, fields such as biomedical engineering, which before had 14 students rank it at the top, now only had 6 rank it the most in need of ethics. This shift, when viewed macroscopically, looks more evenly distributed across majors than before, indicating that the students are seeing a need for ethics across the disciplines.

In the other questions, student tended to answer in the affirmative concerning questions of the need for ethics in general. For example, for questions on whether ethics has an important role to play in software engineering, students answered more towards the “strongly agree” than before, with almost no students answering in “mostly agree” on the second iteration. This trend

continued for the question on whether “Ethics and professionalism go hand in hand” and “Continuing education is a necessary part of being a professional”.

The students also sympathized more with the issue of access in the second survey, with a shifting distribution for the question of software accessibility. They also moved towards agreeance on the question of accessibility of software to those of lower incomes, and further disagreed that the needs of their employer outweighed those of the public good. Students also shifted their views on copying software for personal use without repercussions towards “strongly agree” (i.e., fair use). Also, they shifted towards disagreeing on the topic of whether or not end users have the right to know everything a piece of software does to a computer. In general, students shifted more towards protections for code producers in all questions related to source code and code design, relating the subtle interplay between fair use and intellectual property protections.

5. Summary of the modules and associated pedagogy

A detailed list of the modules as well as all lecture notes, exercises, and assignments can be found online[9]. The modules consist of 5 separate sections covering: (1) basic professional ethics; (2) the software engineering code of ethics; (3) legal issues and security concerns; (4) local and global impacts; and (5) professional development. The general pattern we use for each module is as follows.

1. Introduce basic ideas, terminology, background
2. Look at and discuss case studies (where appropriate)
3. Reflective exercises (individually and in small groups or pairs)
4. Discussion
5. Application exercises (where students apply ideas)
6. Individual writing assignments (tie material to their capstone project and used for assessment)

The modules are designed to take anywhere from 1-3 one-hour lecture periods, where each lecture consists of lecture notes (covering item 1 above) and an exercise sheet (with questions related to items 2-5 above). In our courses, we have students hand in exercise sheets, which are graded and commented on by the instructor. Written homework is assigned at the end of the modules and used in our program for ABET assessment. As mentioned above, the exercises are drawn from a number of sources. The following table describes the various topics and associated exercises.

Area	Topics	Exercise Themes
Software Engineering Ethics I (adapted from Valor & Narayanan[8])	Basic notions of ethics and ethics applied to software engineering. Reasons software engineers should take ethics seriously. Moral harms and public goods.	Reflection on ethics in software engineering. Case study analysis. Software engineers contributions to the public good.
Software Engineering Ethics II (adapted from Vallor & Narayanan[8])	What it means to be a professional. Habits of mind and action.	Reflections on habits of thinking. Exemplars of moral excellence. Applications to the context of software engineering. What an ethical life of a software engineer might be.
Software Engineering Code of Ethics	Types of professional codes. The ACM/IEEE Software Engineering Code of Ethics.	Examination of specific clauses. Development of case studies around specific groups of clauses.
Legal Issues and Responsibilities	Basic notions of liabilities and protections applied to software engineering. Buyer beware vs software maker responsibility debate. Software copyright and patents. Security and privacy issues.	Reflections on liability, and especially whether software creators should be responsible for damages. Security programming practices. Reflections on their own products. Licensing models. Patent searches related to their capstone projects.
Impacts of Technology	Technological neutrality vs determinism. The technical code. Types of social changes (impacts) affected by technology. Empathy. Exercising your moral imagination.	Analyzing technology based on the idea of the technical code.
Professional Development	Revisiting the need for ongoing professional development. Processes for learning. Information sources. The notion of "authoritative" sources and judging whether a source is "good" and/or "effective" for learning. The benefit of side projects in computer science. Rules for lifelong learning.	Applying the learning process. Reflection on resources used within senior design and evaluating whether the source was "good" and/or "effective". Side project brainstorming.

The first written assignment asks students to identify and explain potential professional, ethical, legal, security, and social issues and responsibilities relevant to software engineering within two different case studies. The second assignment asks students to write an essay on the potential societal impacts of the software they are developing as part of their senior-design capstone projects. Students are asked to use notions from the technical code (i.e., both implicit and explicit values and assumptions), examine both negative and positive impacts, and identify and describe who is potentially impacted. Additionally, students must identify and explain three possible modifications of their software to increase the positive and decrease the negative potential impacts. The third assignment asks students to write an essay that explains and analyzes the need for professional software engineers to be aware of and learn new development techniques, tools, and programming languages throughout their careers. Students must relate their analysis to their experiences within their senior-design capstone projects (e.g., what new skills did they learn, how did they learn them, and how did these learning approaches work for

them), and reflect on how they plan to participate in lifelong learning activities after they graduate. Each assignment also was developed with an associated grading rubric that can be used to help determine whether the student is below expectations, is progressing towards mastery, meets expectations, or exceeds expectations.

6. Future changes to our approach and work in progress

We are currently finishing our second year of using the ethics modules within the senior design capstone. As mentioned previously, we are currently in the process of collecting post-survey data to compare with our pre-survey results. Although we have only been using an integrated approach for two years, the results have been very positive (in terms of student feedback and assessment results). Many of the issues surrounding ethics come naturally to students within the context of their senior capstone projects. Students are able to relate topics directly to their projects, including many of the principles associated with (project) management, their relationship with the client and employer, and their relationship with the public from the ACM/IEEE Code of Ethics. However, not only is it easy for students to quickly understand and contextualize many of the topics (since they have experienced them within their senior projects or through other experiences such as internships), discussing broader societal impacts can have a direct influence on their projects within the second semester of the program. For example, students often have a difficult time separating the needs of their client (or their sponsor) from the needs of the end users of their products. However, the success of a product is typically determined on whether the product is useful for the end user, and thus, as a software engineer, understanding the needs of users (and how they may differ to what is being asked by the client or sponsor) is paramount. Studying the impacts of technology on society can help students develop a clearer understanding of the importance of gaining insight into end user (as opposed to just business or technological) needs, which has lead students to better prioritize their efforts on improving their usability testing plans and to obtain additional feedback from potential end users on the quality and usefulness of their product.

In terms of future work, we plan to further expand the modules in the coming year and also to use them earlier within the capstone program (starting in the first semester, as opposed to the second semester of the sequence). In particular, by moving the ethics content earlier in the capstone sequence, we will have an opportunity to place a greater emphasis on the use of ethical considerations to help shape the (user-centered) design choices students make for their projects. Just as empathy and value creation play a large role in design thinking processes, incorporating ethical considerations into the design process can have a positive impact on software design considerations[11, 12, 13]. We feel that this change will not only allow ethical topics to continue to be contextualized and made tangible for students through their senior capstone projects, but that in addition, studying ethics (related to their projects) earlier in the sequence will have the potential

to improve the design and products created by the students. In a similar way, discussing the principles surrounding the ACM/IEEE Code of Ethics, legal and security issues, and professional development within the first semester of the capstone program will allow students to act on issues concerning the topics earlier in their capstone projects. To accommodate this change, we are adding an additional credit to the senior-design capstone course in the Fall for the ethics modules. This will mean that one-hour per week will be dedicated on the ethics content within the entire first semester of the sequence.

Finally, we hope to collect data to better assess students' understanding of professional ethics before and after working through the modules. As part of this effort, we have developed a discipline-specific survey to assess our students' current views on ethics (taken broadly) within computer science (with preliminary results of this pre-survey presented in Section 4). We are currently in the process of performing a similar post-survey from which we can compare before and after responses. We plan to do a similar survey next year, expanded to include additional before and after data on the influences of studying ethics on design. According to Goldweber et al.[1], "the efficacy of strategies for introducing professional and social issues into the [computing] curriculum is not well understood". While few computer science programs integrate ethics within their capstone programs, we hope our strategies can be reused by other programs and that our ongoing assessment of these strategies can contribute to the broader goal of developing approaches for incorporating ethics into computing education.

References

[1] M. Godweber, R. Davoli, J. Currie Little, C. Riedesel, H. Walker, G. Cross, B.R. von Konsky, "Enhancing the Social Issues Components in our Computing Curriculum: Computing for the Social Good", ACM InRoads, 2(1):64-82, 2011.

[2] M.J. Quinn, "On Teaching Computer Ethics within a Computer Science Department", Science and Engineering Ethics, 12(2):335-343, 2006.

[3] P. Barroso, G. Melara, "Teaching of Computer Ethics at the State of California's Universities and other Countries", ETHICOMP E-Journal, 1(3), 2004.

[4] C. Spradling, L.K. Soh, C.J. Ansorage, "A Comprehensive Survey on the Status of Social and Professional Issues in United States Undergraduate Computer Science Programs and Recommendations", Computer Science Education, 19(3):137-153, 2009.

[5] C. Martin, "The Case for Integrating Ethical and Social Impact into the Computer Science Curriculum", Proc. of the ACM Conference on Integrating Technology into Computer Science Education, pp. 114-120, 1997.

[6] ACM/IEEE Software Engineering Code of Ethics. URL: http://www.acm.org/about/se-code

[7] B.G. Blundell, L.W. Liu, “Ethical and Professional Issues: Transcending the Obstacles to Student Engagement”, Proc. of the International Conference for Process Improvement, Research and Education, 2013.

[8] Vallor and Narayanan, "An Introduction to Software Engineering Ethics", Markkula Center for Applied Ethics at Santa Clara University. URL: https://www.scu.edu/media/ethics-center/technology-ethics/Students.pdf

[9] http://www.cs.gonzaga.edu/~bowers/ethics-2015/.

[10] S. Baase, “A Gift of Fire: Social, Legal, and Ethical Issues for Computing Technology (4th Ed.)”, Prentice Hall, 2013.

[11] K. Shilton, S. Anderson, “Blended, Not Bossy: Ethics, Roles, Responsibilities and Expertise in Design”, Interacting with Computers, 2016.

[12] K. Shilton, “Values Levers in Design”, Proc. of the Intl. Conf. on Human Factors in Computing (CHI), 2012.

[13] C. Knobel, G.C. Bowker, “Values in Design”, Communications of the ACM, 54(7):26-28, 2011.