Maintenance Courses in a Software Engineering Curriculum

James Pearson

CSC 300: Professional Responsibilities

Section 1

Dr. Clark Turner

March 8, 2012

Abstract

Although software maintenance makes up most of the work a software engineer will do in the workplace, Cal Poly requires Software Engineering undergraduates to take only two courses that cover software maintenance. Considering this, is it ethical for the Computer Science department to state that the program is designed to produce software professionals? The ACM's Software Engineering Code of Ethics lists maintenance as one of the primary focuses for a Software Engineer; given this, the department should require a more thorough instruction in the art of software maintenance for Software Engineering students.

Contents

1 Facts

The Cal Poly catalog states that the Software Engineering undergraduate degree "prepares students to become software professionals who develop software products on time, within budget, and that meet customer requirements". [2] It also says that the program differs from the similar Computer Science undergraduate degree program in four ways, one of which is having "classes [that] include significant learning in engineering and management areas such as quality assurance, testing, metrics, maintenance, configuration management and interpersonal management skills." [2]

CSC 309's course description refers to "maintenance of large software systems", while CSC 406's simply mentions "software maintenance".[1] [3]

George E. Stark claims that "maintenance consumes between 60 percent and 80 percent of a typical product's total software lifecycle expenditures". [13] Girish Parekh says that it is two-thirds of the lifecycle and consumes at least 50 percent of most programmers' time. [22]

Andrews and Lutfiyya, while preparing in 1998 for a software maintenance course at the University of Western Ontario, found no courses at other universities in which students performed maintenance activities on legacy software. [24] They also found that "most computer science programs offer[ed] no more than two software engineering course". [24]

2 Research Question

Does the Cal Poly Software Engineering program include enough training in software maintenance in CSC 309 and CSC 406?

Cal Poly requires Software Engineering undergraduates to take only two courses whose course descriptions cover software maintenance, and those courses are designed to teach about several other topics as well. [3] However, the ACM Software Engineering code of ethics lists software maintenance as one of the roles of a software engineer [20], and studies have shown that software maintenance consumes a large portion of the software life cycle. [13] [22]

3 Extant Arguments

3.1 Arguments For

Andrews and Lutfiyya found that their software maintenance course "gave students valuable experience in the qualitatively different task of software maintenance". [24] Engle, Ford and Korson state that it is "important for students to have experienced [software maintenance]". [21]

[1] Three courses in the Computer Science department's section of the 2011-13 Cal Poly Catalog have the word "maintenance" in their course description. [3] In one of these, CSC 358 Computer System Administration, the reference is to "system maintenance". [3] Given the course title and that the course is billed as teaching "fundamental concepts of Unix system administration", [3] we can conclude that the maintenance described is of an operating system and its components, rather than a single piece of software in which the maintainer is altering code. Thus, this course is not of relevance to our discussion.

3.2 Arguments Against

3.2.1 Difficulty of Classroom Instruction

Engle, Ford and Korson's project stemmed from a perceived "difficulty of preparing a software system upon which maintenance can be performed"; their end product consisted of 10,000 lines of Ada code and 9 documents. [21]

Andrews and Lutfiyya avoided creating a large maintenance-ready system by involving several clients and working on free software projects under the GNU Project. One student team had both their first and second projects cancelled due to a lack of available time on the part of the client; this meant they had to switch to new projects twice during the term. [24]

In addition, Andrews and Lutfiyya faced complications in grading; each group was working on a different project and each had its own goals. [24] Grades could not even be fairly assigned based on the clients' satisfaction, as one group unknowingly implemented a feature that had been previously finished (but without an updated status in the project wiki). [24]

Andrews and Lutfiyya also noted issues with finding available projects that would be stable enough that student code would not be irrelevant by the end of the course, yet new enough to be using cutting-edge technologies that the students were enthusiastic about. [24]

3.2.2 Cost of Increasing Maintenance Coverage in Existing Courses

One of the Cal Poly courses that covers software maintenance, CPE 309, also covers several other topics:

> Methods and tools for the implementation, integration, testing and maintenance of large software systems. Software development and test environments. Software quality assurance. [3]

CPE 406, the other course, similarly covers a range of topics:

> Deployment of a sizeable software product by a student team. Software maintenance and deployment economic issues. Management of deployed software: version control, defect tracking and technical support. [3]

According to our current understanding of mathematics, increasing the coverage of one subject while holding the total time constant necessitates that time must be taken away from the other subjects. That is, adjusting the breakdown of a currently-offered course without adding more units will always be a tradeoff - an increase in one subject corresponds to a decrease in another. If the department does not wish to decrease time spent on any other subjects, they cannot increase the time spent on software maintenance without adding another course.

3.2.3 Cost of a New Course

The Cal State University system has faced continuous budget cuts over the last decade, reaching its lowest level of funding since 1998. [11] Correspondingly, yearly fees for an in-state student rose from $2976 in 2002 to over $7900 for students entering in Fall

2011 [6] and there has been an almost 9% decrease in the number of CSU employees since 2008. [11]

Cal Poly students have had difficulty in registering for their classes as it is. [23] Adding an additional course requires the department to either replace a section of another course (making that other course more difficult to enroll in) or find the resources for another room and professor.

Cal Poly has not released any statistics about the cost of a course; we do know, however, the salaries of professors. For instance, Dr. Turner received a salary of $99,672.00 in 2011. [18] From scheduling information, he appears to teach 5 4-unit courses a year. [7] This places an estimate of roughly $20,000 for a course he teaches. In addition, the school must arrange access to a room, provide it with electricity, and perform any necessary maintenance and cleaning.

4 Analysis

4.1 Applicability of the SE Code

This paper will use the ACM's Software Engineering Code of Ethics as a basis for evaluating the ethicality of the Computer Science department's course offerings. Therefore, we must first show that the SE Code may be applied to the department.

The preamble to the Code explicitly states that it applies to those engaged in the education of software engineers:

> "Software engineers are those who contribute by direct participation or by teaching, to the analysis, specification, design, development, certification, maintenance and testing of software systems". [20]

A few sentences later, it reaffirms this:

> "The Code contains eight Principles related to the behavior of and decisions made by professional software engineers, including [...] educators". [20]

"The Computer Science Department educates students in the discipline of computer science" and has an "educational mission". [2] Although we will make no claim that computer science is the same as software engineering, the department also claims that "[t]he BS in Software Engineering prepares students to become software professionals". [2] Therefore, the Computer Science department of Cal Poly falls under the educator category of those the SE Code wishes to regulate.

4.2 Accurate Descriptions of Employment

Section 5 of the SE Code states:

> "those managing or leading software engineers shall ... [a]ttract potential software engineers only by full and accurate description of the conditions of employment." [20]

4.2.1 School vs. Employment

Teachers have said that students should "think of school as their job". [28] If students are employees, are the teachers their managers? Both teachers and managers

hand out assignments; both teachers and managers give out grades based on performance. [27] This is an artificial construct (the department is not truly an employer to its students, aside from any extra-curricular employments), but it provides a useful mindset from which to evaluate Section 5.06 for the scope of this paper.

In addition to managers, the SE Code says that it applies to those persons who are "leading software engineers". The Computer Science department has set out a list of courses that Software Engineering undergraduate students must take to graduate. [4] In this way, they are leading students down a specified path of instruction in the field of Software Engineering and thus Section 5 of the SE Code applies to them.

Once again considering the "school is a job" adage, we can draw parallels between the "employment" discussed in section 5.06 and a student's time at a university.

Substituting our terms into the Code, Section 5.06 becomes

> Cal Poly's Computer Science Department shall attract potential software engineering students only by full and accurate description of the conditions of enrollment.

4.2.2 Accurate Description

As stated previously, the catalog descriptions of the Software Engineering program state that a graduate will have experienced "significant learning" in maintenance activities. [2] As "significant" means "sufficiently great or important to be worthy of attention; noteworthy" [14], it is difficult to pin down whether or not any student has gotten "significant learning" in *anything*, yet we will attempt to do so.

As previously discussed, there are two courses offered by the Computer Science department that we must consider when looking at an undergraduate's education in software maintenance. These two courses are "CSC 309 Software Engineering II" and "CSC 406 Software Deployment".

CSC 309 is a "continuation of the software lifecycle. Methods and tools for the implementation, integration, testing and maintenance of large software systems. Software development and test environments. Software quality assurance. Group laboratory project. Technical presentation methods and practice." [3]

From this description, it appears that software maintenance is one of six core concepts covered in the course (implementation, integration, testing, maintenance, development and test environments, quality assurance). In the longest quarter of the year at Cal Poly, there are 51 instructional days. [1] Assuming an even spread of topic coverage, that gives less than 9 days of instruction about software maintenance. In reality, the number is even less, since the course is usually taught for 2 or 3 days a week. [8]

CSC 406 covers "deployment of a sizeable software product by a student team. Software maintenance and deployment economic issues. Management of deployed software: version control, defect tracking and technical support." [3] Again, we see that software maintenance is one of six primary topics addressed in a course taught three days a week. [9]

With only 30 Monday-Wednesday-Friday instructional days in the quarter [1],

a Software Engineering student will receive approximately 10 days of instruction in software maintenance over the course of their academic career. When compared to the total number of years spent at Cal Poly, this number does not appear to be "noteworthy".

As mentioned previously, professional studies have estimated 50-80 percent of a Software Engineer's working time is spent working on software maintenance. [13] [22]

Using the most conservative end of this estimation, we can say that the percentage of working time spent in software maintenance by professional Software Engineers is *over 36 times more* than that spent in instruction in the Software Engineering program at Cal Poly.

Given these facts, it appears the department is not accurate in stating that Software Engineering students experience "significant learning" in software maintenance.

4.2.3 Ethical Analysis

If we can consider the Computer Science department to be equivalent to a student's manager or leader and the term of their enrollment to be employment, the department violates section 5.06 of the Software Engineering Code of Ethics by not accurately representing the amount of software maintenance done by students in the Software Engineering program.

A rule utilitarian will hold that Computer Science departments like Cal Poly's should hold to the SE Code when they can; regular breaking of the rules leads to societal distrust of the profession, which produces less happiness for both Software Engineers and those distrusting them.

In addition, section 1.6 of the SE code tells Software Engineers to

> Be fair and avoid deception in all statements, particularly public ones, concerning software or related documents, methods and tools. [20]

The courses required for a Software Engineering undergrad are part of a software engineering teaching method; thus, section 1.6 commands the Computer Science department to avoid deception when making statements about its Software Engineering program.

If, as previously determined, the program is not providing "significant learning" in software maintenance, as the department claims, the Computer Science department is making deceptive statements to current and prospective students about the Software Engineering program.

Section 1 of the Code regards "the public interest" and the subsections contained within are particulars about that subject. [20] Utilitarianism is, in its essence, concerned with the public good. [34] Rule utilitarianism in particular emphasizes conformance to rules unless given a strong reason to not. Therefore, rule utilitarianism would find unethical the Computer Science department's lack of adherence to Section 1.6 and, following, Section 5.06.

4.3 Ensure Qualification

Section 3.04 of the Software Engineering Code states that a professional Software Engineer must

> Ensure that they are qualified for any project on which they work or propose to work by an

Figure 1: Percentage of time spent on maintenance in CPE 309 and CPE 406 (approximate).

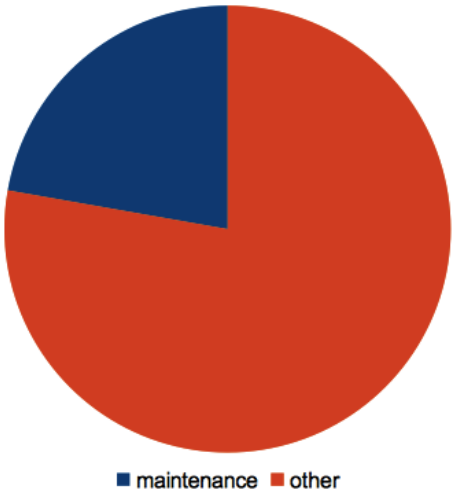


Figure 2: Percentage of time spent on maintenance in Software Engineering major courses (approximate).

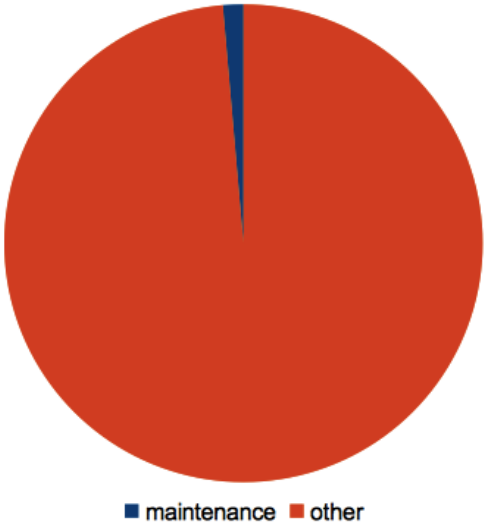


Figure 3: Percentage of time spent on maintenance in Software Engineering all courses (approximate).

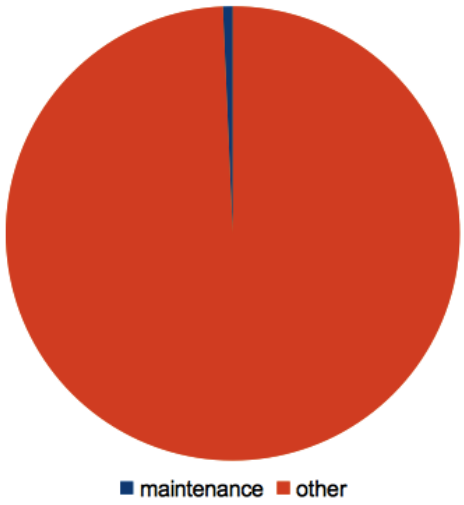
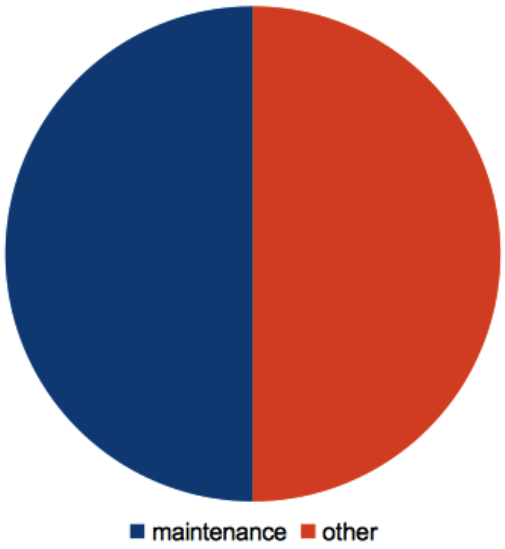


Figure 4: Conservative estimate of working time spent in software maintenance as a Software Engineer.



> appropriate combination of education and training, and experience. [20]

This section is the first building block to analyzing the Computer Science department's behavior according to Section 5.11 of the Software Engineering Code. To this end, it will show that Cal Poly's Software Engineering students are not encouraged or assisted in meeting this standard in regards to software maintenance.

We ask the reader to please remain patient as this multi-layered argument is established.

4.3.1 Applicability of the Code

The Computer Science Department claims that "[t]he BS in Software Engineering prepares students to become software professionals". [2] In addition, Cal Poly Software Engineering alumni frequently go on to pursue professional software development in industry. [5] Therefore, the Software Engineering Code applies to students in the Software Engineering program.

4.3.2 Qualification

A joint group from the Association for Computing Machinery (ACM) and Institute of Electrical and Electronics Engineers (IEEE) compiled a report (entitled "Software Engineering 2004") to "provide guidance to academic institutions and accreditation agencies about what should constitute an undergraduate software engineering education". [19] The report includes recommendations of minimum time to be spent covering certain topics; recognizing that lecture hours are a poor representation of absolute time, given the differences in instructional styles, the authors suggest that the hours should instead be viewed on a relative basis (that is, compared to each other to produce a percentage of time).

Out of the total 494 instructional hours, the report states that at least 10 should be spent on "Software Evolution". The authors equate this to software maintenance:

> Software "maintenance" primarily refers to continued development, or evolution, and not to conventional wear and tear. [19]

In addition, 13 hours are recommended for instruction in "Software Process":

> Software process is concerned with knowledge about the description of commonly used software life-cycle process models and the contents of institutional process standards; definition, implementation, measurement, management, change and improvement of software processes; and *use of a defined process to perform the technical and managerial activities needed for software development and maintenance* (emphasis added). [19]

And 16 hours are recommended for "Software Quality":

> Software quality is a pervasive concept that affects, and is affected by all aspects of software development, support, revision,

> and *maintenance*. It encompasses the quality of work products developed and/or modified [...] (emphasis added) [19]

While giving descriptions of potential course outlines, the authors of Software Engineering 2004 mention that they are basing their work upon a 40-hour course structure. [19] Cal Poly's Software Engineering degree requires roughly 24 courses under the "Major Courses" section [4]; 494 total hours divided by 24 courses gives us 20 hours per course. To correctly compare Software Engineering 2004's hour recommendations with the hours Cal Poly students will experience, we must double SE2004's given hourly units.

As stated previously, we estimate that one-sixth of two standard four-unit courses in a Software Engineering undergraduate degree is spent discussing software maintenance. Assuming again a 40-hour total course instructional time, less than 14 hours are spent discussing the subject.

When examining only the "Software Evolution" section of the report's recommendations, Cal Poly Software Engineering students should be receiving at least 20 hours of time being instructed in software maintenance. In reality, this number should probably be much higher, since some of the 26 hours for "Software Process" and 32 hours for "Software Quality" would also count as instruction in this area.

Given the disparity between actual instructional hours and those recommended by Software Engineering 2004, we can say that students in Cal Poly's Software Engineering program are not qualified for software maintenance activities upon graduation.

4.4 Assist Colleagues in Professional Development

Section 7 of the Software Engineering Code relates to a Software Engineer's behavior in relation to their colleagues:

> 7.02. Assist colleagues in professional development.[20]

4.4.1 Applicability of the Code

The preamble to the Code explicitly states that it applies to those engaged in the education of software engineers:

> Software engineers are those who contribute by direct participation or by teaching, to the analysis, specification, design, development, certification, maintenance and testing of software systems. [20]

Thus, the Code applies to professors in the Computer Science department at Cal Poly.

4.4.2 Colleagues

The Oxford Dictionary defines a colleague as

> a person with whom one works in a profession or business. [15]

As discussed in the previous section 4.2.1 (School vs. Employment), a working parallel regarding school as an employment can be set up for the purposes of this paper. In this understanding, professors are the direct bosses over students. Thus, we can say that students and teachers work with each other in a business.

Also, students are included by the Software Engineering Code as professional Software Engineers:

> The Code contains eight Principles related to the behavior of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, *as well as trainees and students of the profession* (emphasis added). [20]

Therefore, students and professors are colleagues through working in a common profession (software engineering).

4.4.3 Assisting Colleagues

The ACM and IEEE have collaboratively produced "a consensually validated characterization of the bounds of the software engineering discipline" in a document known as the Software Engineering Body of Knowledge (SWEBOK). [12] The SWEBOK consists of 10 knowledge areas, one of which is software maintenance. [12]

As previously discussed in section 4.3 (Ensure Qualification), students in the Software Engineering program are receiving less education in software maintenance than recommended by the Software Engineering 2004 report; this puts them at a disadvantage when they try to fulfill the requirements of the SWEBOK in the years to come.

Therefore, to assist their students in professional development, professors at Cal Poly need to spend more class time covering software maintenance; currently, the professors are not following Software Engineering Code Section 7.02 fully.

4.5 Don't Ask for Code-Inconsistent Behavior

Section 5.11 of the Software Engineering Code states that

> [T]hose managing or leading software engineers shall [...] not ask a software engineer to do anything inconsistent with this Code. [20]

4.5.1 Applicability of the SE Code

As previously discussed in section 4.1 (Applicability of the SE Code), the Code applies in general to the Computer Science department as an educator.

For this particular analysis, we have shown in section 4.4.1 (Applicability of the Code) that the professors in the Computer Science department are considered software engineers under the Code. The department then, as their employer, qualifies for SE Code-applicability as one "managing or leading software engineers". [20]

4.5.2 Asking for Code-Inconsistent Behavior

As discussed in section 4.4 (Assist Colleagues in Professional Development), professors in Cal Poly's Computer Science department are not fully following Section 7.02 of the Code due to not spending enough course time teaching software maintenance to students. By telling professors which courses to teach, the department is then asking them to continue engaging in this non-compliant behavior.

4.5.3 Ethical Analysis

When rewritten to the specifics of our argument, Section 5.11 of the Code becomes

> Cal Poly's Computer Science Department shall not ask professors to teach courses in a way that does not adequately prepare students in the area of software maintenance.

Rule utilitarianism prefers adherence to rules considered to be generally causing the most good. [17] The Software Engineering Code of Ethics was produced and approved by two professional societies, the IEEE and the ACM. [20] If the Computer Science department causes less adherence to the Code through its course offerings, its actions would be considered unethical by a rule utilitarian for breaking a rule that has been agreed upon to help the Software Engineering profession.

5 Conclusion

Cal Poly's Computer Science department claims to prepare Software Engineering students to be professionals in the field of software engineering and to teach them in a number of subjects, including software maintenance. [2] There are a number of arguments as to why the department is currently doing a sufficient job - a maintenance-focused course is difficult to instruct, new courses incur both monetary and educational costs, and restructing an existing course to provide more of a focus on software maintenance would require less time spent on other subjects.

However, a number of professional software engineers have expressed the importance of software maintenance instruction. In addition, best estimates show that the percentage of time spent in the field doing software maintenance as a professional software engineer vastly overshadows the percentage of time a Cal Poly Software Engineering student will spend learning how to do maintenance.

Given these two opposing sets of arguments, we ask the question "Is the Computer Science department spending enough time teaching its Software Engineering students about software maintenance?".

To answer this question, we applied sections 5.06, 1.6, 3.04, 7.02 and 5.11 of the Software Engineering Code of Ethics to information gathered from professional sources and the department itself.

Given the resulting analysis of the department's (lack of) conformance to the Code, we determined using the ethical system of utilitarianism that the department is acting unethically in regards to its current treatment of software maintenance.

References

[1] "2011-12 academic calendar." [Online]. Available: http://www.ess.calpoly.edu/_records/acad_cal/2011_12cal.html

[2] "2011-2013 cal poly catalog." [Online]. Available: http://catalog.calpoly.edu/2011cat/cengr/csc_dept/cscdept.pdf

[3] "2011-2013 cal poly catalog." [Online]. Available: http://catalog.calpoly.edu/2011cat/cengr/csc_dept/csccrs2011.pdf

[4] "2011-2013 cal poly catalog." [Online]. Available: http://catalog.calpoly.edu/2011cat/cengr/csc_dept/softwengr.pdf

[5] "Alums in action!" [Online]. Available: https://www.csc.calpoly.edu/prospective/alums/

[6] "Cal poly san luis obispo - 10 year fee history." [Online]. Available: http://www.calstate.edu/budget/student-fees/fee-rates/sanluisobispo-history.shtml

[7] "Clark s. turner." [Online]. Available: http://schedules.calpoly.edu/person_csturner_last.htm

[8] "Cpe 309 class listing." [Online]. Available: http://schedules.calpoly.edu/classes_CPE-309.htm

[9] "Cpe 406 class listing." [Online]. Available: http://schedules.calpoly.edu/classes_CPE-406.htm

[10] "An examination of the effects of requirements changes on software maintenance releases." [Online]. Available: http://www.mitre.org/work/best_papers/99/skillicorn_releases/skillicorn_releases.pdf

Discusses the effect of maintenance upon schedule

[11] "Following $650 million budget cut, csu trustees approve additional tuition increase for fall." [Online]. Available: http://www.calstate.edu/pa/News/2011/Release/tuitionfall2011.shtml

[12] "Guide to the software engineering body of knowledge." [Online]. Available: http://www.computer.org/portal/web/swebok/

[13] "Measurements to manage software maintenance." [Online]. Available: http://web.archive.org/web/20101112105603/http://www.stsc.hill.af.mil/crosstalk/1997/07/maintenance.asp

Provides citations for sources regarding amount of time spent on software maintenance

[14] "Oxford dictionaries." [Online]. Available: http://oxforddictionaries.com/definition/significant

[15] "Oxford dictionaries." [Online]. Available: http://oxforddictionaries.com/definition/colleague

[16] "Oxford dictionaries." [Online]. Available: http://oxforddictionaries.com/definition/adequate

[17] "Rule utilitarianism." [Online]. Available: http://www.utilitarianism.com/ruleutil.htm

[18] "Search for state worker salaries." [Online]. Available: http://www.sacbee.com/statepay/

Information about state employees' salaries, courtesy of the Freedom of Information Act.

[19] "Software engineering 2004 - curriculum guidelines for undergraduate degree programs in software engineering." [Online]. Available: http://sites.computer.org/ccse/SE2004Volume.pdf

[20] "Software engineering code of ethics and professional practice." [Online]. Available: http://www.acm.org/about/se-code

[21] "Software maintenance exercises for a software engineering project course." [Online]. Available: http://www.sei.cmu.edu/reports/89em001.pdf

Created a 10,000 line Ada program and associated excercises for the purpose of a software maintenance course

[22] "Structured maintenance the warnier/orr way." [Online]. Available: http://books.google.com/books?id=1mQDa_UuMbYC&pg=RA1-PA11&lpg=RA1-PA11

[23] "Students scramble for classes." [Online]. Available: http://mustangdaily.net/students-scramble-for-classes/

[24] J. H. Andrews and H. L. Lutfiyya, "Experience report: A software maintenance project course," in *Proceedings of the 13th Conference on Software Engineering Education & Training*, ser. CSEET '00. Washington, DC, USA: IEEE Computer Society, 2000, pp. 132–. [Online]. Available: http://reversingproject.info/wp-content/uploads/2009/05/experiences_with_a_software_maintenance_project_course.pdf

[25] J. Hughes, "Editorial comments."

[26] B. P. Lientz, E. B. Swanson, and G. E. Tompkins, "Characteristics of application software maintenance," *Commun. ACM*, vol. 21, pp. 466–471, June 1978. [Online]. Available: http://doi.acm.org/10.1145/359511.359522

[27] D. R. McKay, "Employee performance reviews." [Online]. Available: http://careerplanning.about.com/od/performancereview/a/reviews.htm

[28] Z. A. Poerio, "School is a job." [Online]. Available: http://www.nea.org/tools/tips/School-is-a-Job.html

[29] J. Slimick, "An undergraduate course in software maintenance and enhancement," in *Proceedings of the 10th Conference on Software Engineering Education and Training*, ser. CSEET '97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 61–. [Online]. Available: http://ieeexplore.ieee.org.ezproxy.lib.calpoly.edu/stamp/stamp.jsp?tp=&arnumber=592440

Ran a software maintenance course in 1994 and 1996

[30] C. S. Turner, "Feedback on term paper proposal."

Used completely his reworking of my focus question. Notes on extant arguments used as a starting point for further research.

[31] J. Vogel, "Six ways to write more comprehensible code." [Online]. Available: http://www.ibm.com/developerworks/linux/library/l-clear-code/

[32] J. Worthington, "Two faces of software engineering programs."

Used as a structural model for some sections.

[33] S. S. Yau and J. J.-P. Tsai, "A survey of software design techniques," *IEEE Trans. Softw. Eng.*, vol. 12, pp. 713–721, June 1986. [Online]. Available: http://dl.acm.org/citation.cfm?id=5980.5985

[34] B. Zunjic, "Utilitarianism." [Online]. Available: http://www.uri.edu/personal/szunjic/philos/util.htm