Note to Self: Make Assignments Meaningful

Lucas Layman[1] Laurie Williams[1] Kelli Slaten[2]
North Carolina State University
[1]Department of Computer Science, [2]Department of Math Education
Campus Box 8206, Raleigh, NC 27695
+1-919-513-5082
[lmlayma2, lawilli3, kmslaten]@ncsu.edu

ABSTRACT

In addition to "learning by doing," programming assignments and projects are also the mechanism by which students learn about the utility of computer science – or not. Recent research indicates that the current generation of students is in search of a career with meaning, and women and minorities have long been known to desire careers that help society. In this paper, we provide student testimonials on the importance and benefits of practical and socially-relevant assignments. We then examined approximately 200 first year (CS1) and software engineering assignments at top computer science institutions. Only 34% of the CS1 projects had a practical or socially-relevant context, 41% had no context at all, and 15% were games. For software engineering projects, 62% were practical or socially-relevant, but still 16% had no practical context. We recommend that educators, through their assignments, place increased emphasis on demonstrating that computer science can be used to aid society and/or produce products of practical value to society.

Categories and Subject Descriptors

K.3.2. [Computing Milieux]: Computing and Information Sciences Education – *computer science education*

General Terms: Human Factors

Keywords

CS1; Software engineering education; programming assignments

1. INTRODUCTION

. . . ensuring science and technology are considered in their social context with assessment of their benefits for the environment and human beings may be the most important change that can be made to science teaching for all people, both male and female. [21]

The first step in creating a programming assignment is to come up with an idea for the program. The idea should demonstrate the learning objective of the assignment *and* be interesting to the student. In the quest to make assignments interesting, we might forget that impressionable students are forming their opinions on the utility of computer science during the long hours they spend on these assignments. Consider a hypothetical student, Kimberly, in her first year of college. Earlier today in biology, she learned about DNA and its role in detecting genetic diseases. Her next class was English 101; as much as she hates to write essays, she understands writing is an important skill for life. Now, she will spend the rest of her day in the computer lab working on her pie throwing simulation game. The first and second authors are guilty of creating this pie throwing simulation program.

As educators, we understand that the fundamental lessons that underlie our assignments (such as the pie throwing program) could be used in software that could help cure AIDS someday. But can students see past the domain of our assignments to all the great things they can do with computers? Or, are students, particularly those just starting in computer science, left with the opinion that computers are consistently used for useless things like throwing pies in a virtual world? Consequently, are students leaving computer science in search of a major that has a better chance of giving them a career with a more noble purpose?

Women [13] and minorities [8] are known to desire careers that help society. Recent sociological research has indicated that all Millennial students (those born after 1982), including men, are in search of a career with meaning [16]. *The purpose of this paper is to stress the importance for assignments to demonstrate how computer science can be used to aid society or to have some practical value.* To understand students' opinions about the content of their assignments, we conducted over 50 interviews, primarily with women and minority students, and collected hundreds of course reflection questionnaires. Additionally, we examined 170 CS1 assignments and 20 software engineering projects given to undergraduates at top computer science schools to examine their practical and societal value.

The remainder of the paper is organized as follows: Section 2 discusses background on meaningful assignments; Section 3 presents findings that substantiate the students' desires for meaningful assignments; Section 4 presents the analysis of numerous CS1 assignments and software engineering projects. We conclude with recommendations in Section 5.

2. RELATED WORK

This section provides background on current perceptions of the important attributes of computer science assignments and on the preferences of current day students.

2.1 Nifty Assignments

Over the past seven years, Nick Parlante of Stanford University has well served the computer science education community by collecting and distributing successful "nifty" assignments from educators. Each year, six or seven of the top assignments have been presented at SIGCSE, and the support materials for these assignments have been made freely available on a website [18]. We believe the first two criteria of a Nifty Assignment (as enumerated below) embody the prevailing criteria used by educators as they create and/or choose their assignments:

- **Nifty** – the Nifty Assignments often have a playful sort of "fun factor" to them. They are very visual, or they build a game, or they have entertaining output. The assignments invite the students to play around with the material.
- **Topical** – most Nifty Assignments fit into the curriculum and difficulty range that makes sense for most schools (typically CS0-CS2). This is just a practical bias, where we want to promote assignments that can work for the greatest number of students.
- **Scalable** – many Nifty Assignments operate at two levels. First and most importantly, there's the mainstream part of the assignment that is nifty, meaningful, and effective for the average student. Beyond that, many Nifty Assignments have an open-ended aspect where advanced students can take the assignment beyond its original boundaries.
- **Adoptable** – for an ideal Nifty Assignment, the author has put together materials that make the assignment easy for another instructor to adopt: handouts (.rtf, .doc, or .html formats), starter source code, data files, and other ancillary materials.

We believe an additional criterion – meaningful – should be given priority in our assignments. Whereas "nifty" seeks to engage the student by being fun or "cool," a "meaningful" assignment seeks to engage the student by appealing to their practical nature and/or their desire to help society. Certainly, assignments can be both nifty and meaningful. Two such examples on the Nifty Assignments webpage are Name Surfer by Nick Parlante [17] and Natural Prestidigitation [23] by Steve Wolfman.

2.2 Service Learning

Service learning is a pedagogy that integrates academic learning with community-based work. The Engineering Projects in Community Service (EPICS) program [5], initiated at Purdue University, is a project in which multidisciplinary teams, consisting of freshman, sophomores, juniors, seniors and some graduate advisors work together on community projects. Through the Engineers in Technical Opportunities of Service Learning (ETHOS) [4], students at the University of Dayton travel abroad to engage in engineering projects to aid people in other nations. Service learning attracts underrepresented groups to engineering through the context of community-based projects [15]. In general, service learning is difficult to integrate into the core curriculum, particularly with beginning undergraduates. However, even small programming assignments can be tied to aiding society or having practical value.

Since 2002, the capstone software engineering project at University at Buffalo SUNY has involved a socially-relevant project. Some example projects include building an augmentative communications device for a 42-year old stroke patient, and software to help children with cerebral palsy and other disabilities. Students have responded with intensity and philanthropy, indicating that, through the project, not only did they learn about technology, but they were able to give to their community and to expand their world view [2].

2.3 Preferences of Current Day Students

... [a female student] scrutinizes the worth of each computing project in terms of what it is doing to change and help the world. [13]

Sociologists have found that women feel compelled to find a means of serving others and work at this all their lives; doing so makes them comfortable and satisfied. [13] As a result, female computer science students desire to connect computing to other fields. In general, working within human and social contexts makes the study of computer science compelling and meaningful for women. [13] Additionally, minority students often want to devote their lives to helping their communities. These students may not regard information technology as a social-conscience field [8]. In contrast, male computer science students have long been considered to be satisfied if they can make the computer, the machine [13], "do something." In a study conducted at Carnegie Mellon, the boys fantasized about extensions of power, often imagining technology that could overpower natural constraints [13]. However, this perspective appears to be changing with the current generation of male students.

The current generation, often called the Millennial generation, as a whole seem to prefer working on broader impact problems, such as addressing an environmental concern or a community problem [16]. The following quote expresses this generation's priorities:

> *Community service is not just an opportunity to the Net Generation, it is a responsibility ... It has become increasingly "cool" to give back ... This acceptance of and emphasis on social responsibility has also changed the way the Net Generation looks at careers. Priority within our ranks is placed less on monetary value and fame than happiness and "doing something good."*
>
> *-- Carie Windham , student [16].*

2.4 The Appeal of Games ... for Some

just ... making video games ... is not ...worth the energy and time that it takes. [13]

Instructors may assume that games are a way to keep students interested and engaged [6], and that students' appeal for playing games translates into their desire to pursue computer science. However, as the quote above indicates, this is not true for all students. Since women are not avid gamers, so they would not be avid game programmers. We do not dispute that games are appealing to some students and even attract some students to the field, but not all students are motivated by games. Games can be fun and engaging, but they are often transient distractions that rely on challenge and entertainment to engage the user rather than more enduring motivations such as self-development or utility.

3. OUR STUDENTS CARE

In Section 2.2, we provided references to the literature indicating that current students are concerned about their ability to contribute socially with computer science. We observe this same phenomenon with our own students. In this section, we briefly discuss the Thinking-Feeling dimension of the Myers-Briggs personality types [11] and its application to meaningful projects. We also provide evidence from interviews with software engineering students conducted over the past three years on the importance of a meaningful project.

3.1 Thinkers and Feelers

The thinking-feeling dimension of the popular Myers-Briggs personality types is highly relevant when considering the meaning and practical value of assignments. Thinkers are rational and logical in their decision making processes, but Feelers make decisions based on intuition and personal consideration. For Feeler students (approximately 25% of software engineering students [3, 12, 14]), human consideration and emphasis on social relevance are particularly important. The work they do must reach beyond a simple learning exercise and have impact beyond determining their grades.

In a previous study on personality types [12], Feeling students commented positively about the practical, socially-relevant software engineering projects they developed. These projects included a bug-tracking system, a software reliability tool, and an information management and web portal system for a state-owned forest. For example, one Feeler student commented,

> *A lot of projects done in school seem to miss on usefulness. However, right from the get go, it was clear the usefulness and importance of our project. I am so satisfied with the out come of the project, that if I was working on something with other people, I would use this [system]. [12]*

Another Feeler student noted,

> *I enjoyed the project mainly because I like programming, but also because I like the idea of working on something so practical and "real-world" as an Eclipse plug-in. [12]*

3.2 Women, Minorities, and Millennials

We have interviewed software engineering students at North Carolina State University, North Carolina A & T (an HBCU), and Meredith College (a women's university) to obtain their reflections on collaborative work [1, 10, 12, 22] and have collected over 50 interviews and hundreds of course reflection questionnaires over the past several years. We have encouraged the software engineering course instructors at the aforementioned institutions to select projects that are socially relevant or at least are practical. The response from students has been positive.

One female student described both the pleasure of working on a practical project (a software development aide) and the pains of working on "mere exercises" in previous classes,

> *I enjoyed the project, mainly because I felt that we were creating something that could actually be used by a customer rather than just an arbitrary program that taught us different aspects of Java like many of the programs that I've written up until now [10].*

An African-American male, commenting on a library reservation system he wrote for his project, observed,

> *… I enjoyed that the most because it's more of a real-world thing. Because it's something that if you were doing it in real life, you'd sit down and do the same thing – you'd go in, log in, search, if the book isn't there, go and try to place it on hold, if it's there, check it out, then you've got to bring it back on time. So, it's just real life, you know. So, I think that's the project – yeah, I'd say I like the most.*

These comments are typical of many students, male and female, minority and non-minority, and Millennial alike – students appreciate, enjoy, and prefer projects that are practical and go beyond simple problem-solving, idea-illustrating exercises.

4. UNIVERSITY PROJECTS

We wanted to examine the current state of affairs in CS1 and software engineering to see just how many assignments given to students exposed them to the practical use and/or social importance of their chosen field. We assessed nearly 200 CS1 and software engineering projects found online to determine the types of assignments we are giving our students.

4.1 Assessing "meaningful" assignments

We selected the institutions in our study from the American Society for Engineering Education's annual listing of the top schools in number of bachelor's degrees awarded in computer science [9]. We visited the websites of approximately 70 computer science programs in search of examples of assignments from introductory CS1 courses and software engineering courses. The information was gathered primarily from course listings and instructor websites. At most, one introductory course and one software engineering course from each institution were used as data points. We selected the most recent course offerings that we could find that provided enough information for our analysis; the oldest samples used were instances of classes taught in Fall 2005. In total, data were collected for 21 CS1 courses and 20 software engineering courses from 30 institutions containing 170 CS1 assignments and 20 software engineering project descriptions.

For the CS1 and software engineering courses, we generated brief descriptions of each programming assignment or project in the course (written assignments were noted but not used in analysis). We were primarily interested in the nature of the programming assignments. Were the assignments either: a) games, b) programs with little practical context (but not games), c) programs with practical context, d) or programs with social relevance?

"Practical context" is a debatable term. We determined if an assignment had practical context by answering the question "hypothetically, could this project have value to an external user?" Certainly applications developed in CS1 might never be used by anyone in actuality. However, there are clearly some assignments (e.g. an employee paycheck program) that convey a practical context with hypothetical real-world value while others (e.g. drawing a Christmas tree using recursion) do not have value to an external user except maybe as mild entertainment. A "socially relevant" assignment had both a practical context and affected, hypothetically, a wider population than a specialized user (e.g. a program used by doctors to dispense medications). The bulk of the categorization is based on the subjective assessment of the first author. In nearly all cases, assignments clearly fell into one of the four categories listed above. When the question of practical context was not clear (e.g. a program that solves the quadratic equation), we placed the assignment in the "practical context" category.

Examples of programs that fall into the Game category were a Hangman-type guessing game or tic-tac-toe. The No Practical Context category was filled with assignments such as "sort this list of integers using bubble and selection sorts," "compute the floor and ceiling of these sets of numbers," and "write a class that randomly generates a coin flip." The commonality among all assignments in the No Practical Context category was that they were designed to illustrate an idea, but did not convey any sense of the utility of the idea. Assignments such as banking systems, employee payrolls, and CD organizers were placed in the Practical Context category since the supposed utility of such applications was directly incorporated into the assignment. Finally, assignments such as a CS1 assignment that involved providing contact information to emergency dispatch personnel and a software engineering project to create a system to manage medical records were considered to be in the Socially Relevant category.

Each assignment or project in all of the courses was assigned to one of the aforementioned categories and a percentage was made on a per course basis. These percentages were then averaged over the entire sample to create a weighted average based on number of assignments per course. In the case of software engineering courses where the term project could fall into several categories based on which project was chosen, an equal percentage was assigned to each applicable category for that course.

4.2 Findings

Table 1 shows the cumulative percentages of assignments/projects that were categorized. We add another column for the popular Karel [19] teaching language, which is not readily classified into any of the aforementioned categories.

Table 1. Average assignments per category (weighted by number of assignments per course)

	Game	*No Practical Context*	*Practical Context*	*Social Relevance*	*Karel*
CS1	15%	41%	34%	2%	9%
SE	23%	16%	32%	30%	0%

We found that an average of 41% of the CS1 assignments (not including games) had no practical context. Further, only an average of 2% (3 of 170 in total) of all CS1 assignments had any sort of social relevance. In the software engineering courses, many projects had some practical context and many were socially relevant. However, an average of 16% had no practical context.

4.3 Discussion

We first focus on the number of assignments in the No Practical Category in CS1 (over 40% on average). Bearing in mind that the samples were taken from institutions that graduate between 70 and 200 computer scientists every year, this implies that a large number of beginning computer scientists are exposed to exercises of the type that one might find in a mathematics textbook transposed into computer programs. We allow that the CS1 courses in our study may have by random chance been atypical of many CS1 courses, though our only criterion for inclusion in the study was sufficient information on the course webpage to categorize each assignment.

One might argue that beginning computer scientists are not capable of producing programs that are complex enough to carry any sort of practical context. However, we found three CS1 courses that *only* contained projects with practical context and one with social relevance! The assignments in one CS1 course used in our study were based entirely on Scheme and were created by Kathi Fisler at the Worcester Polytechnic Institute [7]. The first assignment in this course used a library provided by the instructor with several predefined functions for creating images of fabric, "cutting" clothing items from fabric, and adding logos to fabric and clothing items. The students then use these functions to compose images of clothing items in the fabrics and logos that they designed. Another institution had five CS1 assignments that cumulated in the creation of a simple web crawler. So while CS1 projects may not have real-world *utility*, the instructors were able to create assignments with real world *context*.

Furthermore, many of the assignments in the Practical Context succeeded in creating this context without introducing more complexity into the actual programming than is typical of a CS1 assignment. For example, the first assignment in the CS1 course at the University of Illinois Urbana-Champagne involved computing Student Demographics [20]. The bulk of the assignment involved printing text to a screen, declaring and using variables, performing arithmetic operations on the variables, and getting input from the user through the keyboard. The most complex part of the assignment for a beginning student may be getting information from the keyboard, but a module was provided to the students to hide the complexities of this operation.

We also observed that an average of 15% of the assignments fell into the Game category for CS1 courses. This included puzzle games, word games, number games, and more. In many of the CS1 courses, one or no assignments involved games. However, in three of the courses (14%) over half of the assignments were games. There were four CS1 courses (19%) that had assignments that were only either games or had no practical context.

The software engineering project courses in our study show a more even dispersal among the categories. On average, more than 60% of the projects either had a practical context or were socially relevant, with half that number being socially relevant. These projects included time sheet entry systems, medical record systems and a distance learning applications. Another 23% of the software engineering projects on average were games. In several courses (7 of 20), students proposed and selected their projects, though many courses had projects that were instructor assigned (9 of 20). The remaining courses had a third-party client, typically in the form of a local company or another university department (4 of 20). In three software engineering courses, students were assigned projects of no practical context or games and had no alternative topics from which to choose.

5. CONCLUSIONS

The analysis of both CS1 and software engineering courses assignments is clear: there is much room for improvement in creating "meaningful" nifty assignments. Creating practical assignments for CS1 is often neglected in favor of exercises or games. Creating socially meaningful assignments is also possible, though perhaps overlooked. Changing the context of an assignment may be as simple as changing a few words in the assignment and a few variable names in the program. Instead of a simulation where an adventurer wanders around a maze, create a simulation of a bus service traversing the streets. Instead of throwing pies, analyze the impact of new medicines on patients.

Pie-throwing simulation assignments are like etudes in music. An etude is a short musical composition written solely to improve technique. Musical students are given etudes – short scores of music – to play over and over and over again. These etudes are not pleasing to the ear. The purpose of learning to play etudes is to engrain how to play that kind of a combination of notes. Then later, when that sort of combination of notes appears in the midst of a larger beautiful composition, the notes will just flow off the musician's fingers. Learning each etude is fairly painful and time consuming, but the practice leads to beautiful music. If we embed our fundamental lessons in painful, pie-throwing etudes, can our students see ahead to the beautiful music?

6. ACKNOWLEDGEMENTS

We acknowledge the contributions of Steve Czenszak in the coding of the interviews and examination of student projects at other universities. We would also like to thank Andy Meneely and Mark Sherriff for their helpful comments. This material is based upon the work supported by the National Science Foundation under the Grant No. 00305917. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

7. REFERENCES

[1] S. B. Berenson, K. M. Slaten, L. Williams, and C.-w. Ho, "Voices of Women in a Software Engineering Course: Reflections on Collaboration," *ACM Journal on Educational Resources in Computing*, vol. 4, 2004, pp. 1-18.

[2] M. Buckley, H. Kershner, K. Schindler, C. Alphonce, and J. Braswell, "Benefits of using socially-relevant projects in computer science and engineering education," proceedings of Technical Symposium on Computer Science Education (SIGCSE 2004), Norfolk, VA, 2004, pp. 482-486.

[3] L. F. Capretz, "Personality Types in Software Engineering," *International Journal of Human-Computer Studies*, vol. 58, 2003, pp. 207-214.

[4] "University of Dayton International Programs / ETHOS," http://international.udayton.edu/edabroad/students/programs/ethos.htm, University of Dayton, accessed November 15, 2006.

[5] "EPICS at Purdue University," http://epics.ecn.purdue.edu/, Purdue University, accessed November 15, 2006.

[6] M. Feldgen and O. Clua, "Games as a motivation for freshman students learn programming," proceedings of ASEE/IEEE Frontiers in Education, Savannah, GA, 2004, pp. S1H11-S1H16.

[7] K. Fisler, "WPI CS 1101 A05 Home Page," http://web.cs.wpi.edu/~cs1101/a05/, Department of Computer Science, Worcester Polytechnic Institute, accessed September 6, 2006.

[8] P. Freeman and W. Aspray, "The Supply of Information Technology Workers in the United State," Computing Research Association, Washington, DC, 1999.

[9] M. T. Gibbons, "The Year in Numbers," American Society for Engineering Education, Washington, DC, 2005.

[10] C.-w. Ho, K. Slaten, L. Williams, and S. Berenson, "Examining the Impact of Pair Programming on Female Students," North Carolina State University Department of Computer Science, Raleigh, NC, TR-2004-20, June 17, 2004, 2004.

[11] G. Lawrence, *People Types and Tiger Stripes*, 3rd ed., Center for Applications of Psychological Types, Gainesville, FL, 1994.

[12] L. Layman, T. Cornwell, and L. Williams, "Personality Types, Learning Styles, and an Agile Approach to Software Engineering Education," proceedings of ACM Technical Symposium on Computer Science Education (SIGCSE '06), Houston, TX, 2006, pp. 428-432.

[13] J. Margolis and A. Fisher, *Unlocking the Clubhouse: Women in Computing*, The MIT Press, Cambridge, Massachusetts, 2002.

[14] M. H. McCaulley, "The MBTI and Individual Pathways in Engineering Design," *Journal of Engineering Education*, vol. 80, 1990, pp. 537-542.

[15] W. Oakes, J. Duffy, T. Jacobius, P. Linos, S. Lord, W. W. Schultz, and A. Smith, "Service-Learning in Engineering," proceedings of ASEE/IEEE Frontiers in Education, Boston, MA, 2002, pp. F3A-1 - F3A-6.

[16] D. Oblinger and J. Oblinger, "Educating the Net Generation." Boulder, CO: Educause, 2005.

[17] N. Parlante, "Name Surfer - Nifty Assignments," http://nifty.stanford.edu/2005/NameSurfer/, Stanford, accessed September 6, 2006.

[18] N. Parlante, "Nifty Assignments," http://nifty.stanford.edu/, Stanford, accessed September 6, 2006.

[19] R. E. Pattis, *Karel the Robot: a Gentle Introduction to the Art of Programming*, 2nd ed., Wiley, New York, 1995.

[20] C. Peiper, "CS 125 Home Page," http://www.cs.uiuc.edu/class/fa06/cs125/, Department of Computer Science, University of Illinois at Urbana-Champaign, accessed September 6, 2006.

[21] S. Rosser, *Female-friendly science: Applying women's studies methods and theories to attack students*, Pergamon Press, New York, 1990.

[22] K. M. Slaten, M. Droujkova, S. Berenson, L. Williams, and L. Layman, "Undergraduate Student Perceptions of Pair Programming and Agile Software Methodologies: Verifying a Model of Social Interaction," proceedings of Agile 2005, Denver, CO, 2005, pp. 323-330.

[23] S. Wolfman, "Meta-Commentary on Natural Prestidigitation Assignment," http://nifty.stanford.edu/2006/wolfman-pretid/, Stanford, accessed September 6, 2006.