Engagement and Frustration in Programming Projects

Stuart Hansen
University of Wisconsin - Parkside
900 Wood Rd
Kenosha, WI 53144
1-(262) 595 - 3395
hansen@uwp.edu

Erica Eddy
University of Wisconsin - Parkside
900 Wood Rd
Kenosha, WI 53144
1-(262) 595 - 2734
eddy@uwp.edu

ABSTRACT

Attracting and retaining quality students is an ongoing concern in Computer Science. Retention can be enhanced by keeping students engaged in the learning process while preventing them from becoming overly frustrated. While engaging students during class is certainly important, most students spend a significant amount of time working on programming projects outside of class. The goal of this research was to collect and analyze some initial data on how engaging and frustrating our students find our programming projects. During the Spring 2006 semester we surveyed our students after each programming project. This paper presents some initial findings from the surveys and discusses strategies on how to improve our projects based on this feedback.

Categories and Subject Descriptors

K.3.2 [**Computer and Education**] Computer and Information System Education – *curriculum, self-assessment.*

General Terms

Algorithms, Measurement, Design, Experimentation, Human Factors

Keywords

Programming Projects, Frustration, Engagement

1. INTRODUCTION

Computer Science enrollments have been declining for the past several years. Departments once burgeoning with students now face half empty classrooms. We can no longer rely just on the reputation of our discipline to attract and retain students. It behooves all CS educators to look for ways to attract and retain more quality students in our programs. Common sense tells us that students who find their CS courses engaging will be retained at a higher rate than those who don't. Similarly, students who are overly frustrated by their CS courses may change disciplines.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGCSE'07, March 7–10, 2007, Covington, Kentucky, USA.
Copyright 2007 ACM 1-59593-361-1/07/0003...$5.00.

There is a relationship, too, between engagement and frustration. Margolis and Fisher discuss the relationship between confidence and interest by women studying CS [6]. They document a downward spiral that often occurs where women lose confidence in their abilities which leads to lower interest. We believe the same pattern applies to many students regardless of gender. We want to develop curricula that interests our students and builds a "can do" sort of confidence.

Engagement and frustration are not the same as interest and lack of confidence, but they are related. Engagement is what pulls the student to a particular activity, or what interests them now. By contrast, interest reflects more one's relationship to the discipline. A student is engaged by a programming project, but is interested in computer science. Similarly, frustration is not the same as a lack of confidence, but being overly frustrated can easily lead to a lack of confidence.

Programming and software development are still close to the heart of computer science. McGettrick et al. recognize it as one of the grand challenges in computing education. They claim, "… a strong correlation exists between programming ability and other computing skills, reflecting, as it does, skills in abstraction, conceptualization, design and evaluation [7]." Students spend many hours studying and working on programming projects. If students are overly frustrated by the projects, or even if they are simply not engaged by the projects, they may be turned off to CS entirely. We should be seeking ways to make our programming projects engaging, while reducing the frustration associated with them to a reasonable level. This paper presents a first step, collecting some baseline data, to measure students' reactions to our programming projects.

2. PROGRAMMING PROJECTS

The authors have been collaborating on teaching computer science for years and have a well thought out and consistent approach to developing projects. Obviously, each project is designed to reinforce topics being discussed in class. We also work hard to have a wide variety of problem types represented by the projects, particularly in CS1 and CS2. Our curriculum is not breadth first, but we feel it is important for students to be exposed to a breadth of computer science topics early in their studies. Thus, for example, in CS1 we always try to choose at least one project that involves Monte Carlo simulation, another that involves simple graphics, and yet another that is numerically intensive. This approach helps tweak student interest and reinforces to the students that they are studying computer science and not just programming.

We have also always had implicit goals of creating programming projects that are engaging and not overly frustrating. When we recognized these as explicit goals, it became important for us to measure how successful we are at meeting them.

2.1 Engagement

Dictionary.com gives as synonyms for *engage*: *absorb, engross, interest, involve*[5]. When we defined *engage* to our students, we used the terms: *fun* and *challenging*. We want our students to have fun developing their programs, and at least part of the fun should be because the project is challenging. SIGCSE's Nifty Assignments web site and annual panel discussion follows the same lines [8]. Another way to view our goal is to have all our assignments on a level we would feel comfortable submitting to Nifty Assignments.

2.2 Frustration

Dictionary.com defines *frustration* as: *a feeling of dissatisfaction, often accompanied by anxiety or depression, resulting from unfulfilled needs or unresolved problems* [5]. We recognize that students will feel some frustration while working on projects, but by the time a student submits their assignment, we want the problems to have been resolved and the frustration to have abated.

Our goal is to limit frustration to a reasonable level, where students learn to cope with it successfully. We don't believe it is possible or desirable to remove all frustration from programming. First, students bring a wide variety of backgrounds and skills to our classes. Projects that might be easy for our best students might be overly difficult for weaker students.

Also, computer science is still a rapidly changing field. If we look at the evolution of ACM's Computing Curricula documents, two things stand out: how much the field has changed, and how much the field has expanded [1, 2, 3, 4]. All computer scientists face ongoing pressure to stay professionally current. Because of this, the authors are convinced that a certain level of frustration is endemic in our discipline, and removing all frustration from our courses would not help students prepare for their professional careers.

3. THE SURVEY

During the Spring 2006 semester we surveyed three different classes, CS1, CS2 and, Data Structures and Algorithms (DS), on the due date of each of their programming assignments. The goal of the survey was to collect data showing how our students felt about the projects on our two criteria, engagement and frustration.

We chose CS1, CS2 and, DS because each emphasizes software development skills, including programming, software and algorithm design and testing. All three courses were taught using Java as their language of choice. CS1 was taught using an objects early approach. BlueJ was used as the IDE. CS2 presented more advanced object-oriented concepts, as well as introducing elementary data structures. DS covered more advanced data structures and algorithm design techniques. It gave a mathematical treatment to some of the theoretical topics introduced in earlier courses, like the asymptotic efficiency of algorithms.

3.1 Survey Questions

We designed the survey to be as brief as possible for several reasons. We didn't want the survey to be overly intrusive on class time and we didn't want the survey to be onerous to students. The survey consisted of five short questions, with an opportunity to make further comments at the end. The questions were:

1. `On a scale of 1-10, how engaging (fun/challenging) did you find this assignment?`
2. `What was engaging about it?`
3. `How could it be re-worked to make it even more engaging?`
4. `On a scale of 1 to 10, how frustrating did you find this assignment?`
5. `What did you find most frustrating about the assignment (circle one answer)?`
    a. `Understanding what was required by the assignment`
    b. `Java language syntax and semantics`
    c. `Required knowledge not yet covered in class`
    d. `Designing the classes and methods`
    e. `Working with the program development tools, e.g., BlueJ, netBeans, or emacs`
    f. `Working with Linux`
    g. `Other (please write a short description):`

The survey was administered anonymously. It was also optional, but a large majority of students chose to complete it. The length of the survey made it easy to complete in less than five minutes.

4. SURVEY RESULTS

Table 1 summarizes the survey result by course. When entering the survey data, we had to do some minimal data cleaning. For example, several times students enter values (greatly) exceeding 10 for their frustration level. In this case we assigned 10 as the most appropriate value within the survey range.

Table 1. Summary of Engagement and Frustration by course. n is the number of surveys completed for the course.

	n	**Engage. Mean**	**Engage. S.D.**	**Frust. Mean**	**Frust. S.D.**
CS1	186	6.77	1.97	4.95	2.29
CS2	82	7.56	1.65	5.48	2.12
DS	142	7.59	1.78	5.55	2.19
Total	**410**	**7.21**	**1.89**	**5.26**	**2.23**

4.1 Summary Statistics

We were pleased with the overall Engagement mean of 7.21. Students find their projects engaging. We can look for ways to make each of our assignments more engaging, but the results will be incremental.

We were less pleased with the overall Frustration statistics. The mean of 5.26 is acceptable, but the relatively high standard deviation of 2.23 means that students vary a lot in their frustration levels. This suggests that as a long term goal, controlling frustration is a more major challenge than improving engagement.

These trends were consistent within courses, as well. While there are some differences, courses consistently had higher Engagement scores than Frustration scores and the Frustration standard deviations were consistently higher than the Engagement standard deviations. This means that whatever we are doing right or wrong, we are doing it the same in all our courses and targeting areas for improvement is not course dependent.

4.2 Differences among the Courses

Table 1 suggests that students in CS1 find their programming projects less engaging than students in either CS2 or DS. This result is statistically significant (α=0.01) when comparing CS1 Engagement to either CS2 or DS. There is no statistically significant difference in the means between CS2 and DS.

We should note that CS1 students do find their assignments engaging. A mean value of 6.77 is relatively high on a scale of 1-10. The lower mean for CS1 is also intuitive. Students in CS1 are just learning to program. We are often forced to limit the scope of their projects based on their current background. We do not find the lower CS1 Engagement mean to be a cause for concern.

Table 1 also suggests that students in CS1 find their programming projects less frustrating than students in either CS2 or DS. However, this result is not statistically significant (α=0.05) because of the larger variance in scores.

Another clear difference between courses emerges when we look at Question 5 of the survey which asks students to give the primary source of their frustration. 52 of 160 (32.5%) of CS1 surveys who answered to this question said that ***b. Java language syntax and semantics*** was the cause of their frustration. This was by far the largest group of any of the causes recorded for CS1. It is also intuitive, as CS1 students are learning the fundamentals of programming. It is not clear whether a different language that is cleaner and simpler than Java would address this issue.

In both of the other courses the distribution of frustration causes was much more uniform. The modal group in both CS2 was ***a. Understanding what was required by the assignment,*** with ***b. Java syntax and semantics*** ranking second. We were a bit disappointed in this result, as we work hard to write lucid project descriptions. In DS we often assign projects where applying concepts and algorithms discussed in class are meant as the main challenge. Unsurprisingly, the modal group for frustration causes in DS was ***d. Designing the classes and methods***.

5. PROJECT ANALYSIS

A key goal of this research is to analyze and compare individual projects, to determine which were successful/unsuccessful and why. The assignment means of the two primary numeric variables, Engagement and Frustration, are plotted in Figure 1. Ideally we would like each project to have a high Engagement value and a relatively low Frustration value.

When examining Engagement on a project by project basis we were relatively happy with our results. Except for a single outlier (point A Figure 1) all the projects have mean Engagement values above 6.4. The maximum Engagement mean for a project was 8.43.

The project means for Frustration show a lot of variation. Again, if we ignore the outlier at point A, the range of values is 3.73 to 8.08. As we stated in the previous section, the largest challenge we face in improving our programming projects is reducing the level of frustration. Figure 1 suggests that we should focus on the few projects with extremely high Frustration means (regions B and C in Figure 1). We want to analyze the cause of the students' frustration in those projects, and control it when we reuse those assignments or similar ones.

5.1 No Correlation between Frustration and Engagement

When we began the experiment, we naively expected an inverse correlation between Engagement and Frustration. That is, we expected that the more engaged students were by a project, the lower their frustration levels would be. As Figure 1 shows, however, there is no significant correlation between the variables. The only noticeable pattern is that the Frustration outliers occur more frequently (points in regions A and B in Figure 1) when Engagement is low.

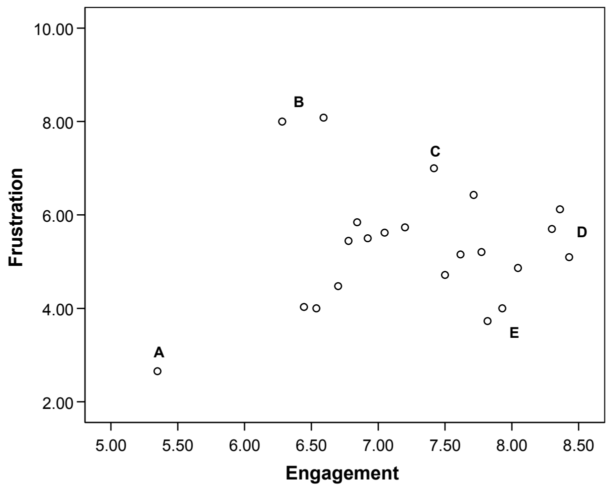

Figure 1. This scatter plot shows the means of the Engagement and Frustration scores for each of the 22 projects in the survey. The collections of points labeled A-E represent programming projects with particularly high or low scores.

5.2 Explaining the Outliers

We have labeled collections of points A-E around the edge of the graph in Figure 1. Points in regions A and B represent outliers, separated from the other points by some distance. Regions C, D and E, while not outliers, represent projects that are worth discussion separately. We will discuss each of these collections of points in turn, trying to discern why the projects were particularly good or bad.

5.2.1 Low Frustration – Low Engagement

Point A has the lowest Frustration and the lowest Engagement score of any assignment. It represents the very first programming project in CS1. The goal of the assignment is to familiarize students with BlueJ, including the processes of editing, compiling, debugging and running a program. They are given a hardcopy of a prewritten program and told to enter and run it. It is not surprising that they found this assignment neither engaging nor frustrating. It still teaches important skills and we have no concerns about continuing to use it or similar projects.

5.2.2 *Frustration affecting Engagement*

The two points in region B represent projects that were highly frustrating and not very engaging. We consider these two projects as our failures for the semester.

One of the assignments was a String manipulation/search assignment in CS1. In this assignment students were given a sentence and asked to find and underline a hidden word within it. The hidden word is made up of consecutive alphabetic characters in the sentence, but may be separated by white space or punctuation that is to be ignored. For example, "`When the clothes washer finishes its last` **`spin, a ch`**`ime will sound.`" contains the word "`spinach`". The high frustration scores on this assignment stem from having to maintain data structures quite complex for CS1. Many students were able to create a new string from the sentence with the whitespace and punctuation removed. They could also find the hidden word in the new string. Problems arose, however, when they tried to return to the original sentence and underline the appropriate characters. The start and end indices are not the same in the two strings. Even with very explicit directions, students had problems locating the hidden word in the original string.

The other assignment that fell into region B was from the Data Structures and Algorithms course. This assignment was given early in the semester. It was meant mostly as a review exercise to refresh students Java skills. Students were assigned to write a Java program to find collinear collections of points in a large data set. Students were given a high level discussion of a fast algorithm to accomplish the task and then instructed to implement it. While conceptually the algorithm isn't difficult, there are numerous places where students could (and did) have problems. The high frustration level for this assignment seems to have two roots:

1. It was the first time many students had just been given a problem and an algorithm to solve it and told to implement it. In CS1 and CS2 we typically give more guidance including being told what data structures and methods to implement.
2. The instructor provided sample data sets for the students, but did not provide them with sample solutions. This was the first time that many of the students had to take responsibility for completely testing their own code. The problem was compounded by a bug in the data set generating code that allowed duplicate points within the data sets.

5 of 21 students (23.8%) who responded to question 5 on the survey gave ***g. Other***, as their response and then really let the instructor know about their frustrations in their written comments.

Explaining the low Engagement means for these two projects is more problematic. We hypothesize that the low Engagement scores on these assignments stems from the very high frustration students felt. While overall our data does not show evidence of correlation between these variables, student comments on these assignments lead us to believe that very high frustration does cause decreased engagement.

5.2.3 *Polymorphism is Tough*

Point C in Figure 1 represents the expression tree project in CS2. The expression tree project was the final project of the semester and brought together ideas from throughout the semester including, binary trees, recursion and polymorphism. Expression trees are a nice example illustrating inheritance and polymorphism. An expression tree is built out of internal operator nodes, which contain mathematical operators such as + and *, and external, operand nodes, which contain numeric values. An *Operator* class and an *Operand* class each implement a *Node* interface, which declares methods like `toString()` and `evaluate()`.

Polymorphism was the cause of the high frustration levels for this assignment. 6 of 11 students suggested that their source of frustration related to how Java was handling the recursive calls to various *Node* types.

It is interesting to note that Engagement mean for this project was 7.42, only slightly below the overall CS2 average. This leads us to believe that, unlike the previous section, if the cause of frustration is rooted in understanding complex concepts, rather than the details of a particular problem's solution, students will continue to be engaged, even though frustrated.

5.2.4 *Classic Computer Science is Engaging*

The points in region D of Figure 1 represent projects that were highly engaging. They had Engagement means above 8.3. Two of these assignments were in Data Structures and Algorithms and one was in CS1.

The Huffman encoding project had the single highest Engagement mean (8.43). This is a classic example of a greedy algorithm that builds an optimal encoding of a file using both priority queues and binary trees. At the completion of this assignment, several students asked if this was how modern compression algorithms worked. This led to a brief, but interesting classroom discussion of the Huffman and Lempel-Ziv-Welch (LZW) algorithms. (LZW is covered in our curriculum in the Files and Databases course.) Because they had just coded it, students quickly understood that the Huffman algorithm was a multi-pass algorithm and that it requires the encoding also be stored, and that LZW avoids these problems.

In second place, with an Engagement mean score of 8.36, was an assignment to solve SuDoKu puzzles using a depth first search strategy. SuDoKu is the currently widely popular number puzzle where a person enters the digits 1-9 into a partially completed 2D grid, under certain constraints. When humans solve SuDoKu puzzles, they apply logic rules to find a cell with only one possible value, fill in that value, and then repeat the process. Programming a depth first search solution takes a very different approach. Values are filled in until one of the logical constraints is violated, at which time backtracking occurs. Student comments lead us to believe they found this project engaging both because it involved a popular puzzle and because it was a direct application of an algorithmic approach being studied in class.

The third most engaging assignment, with a mean Engagement score of 8.30, was a Monte Carlo simulation of poker hands. This was the last assignment for the semester in CS1. Students were given two weeks to complete it. Students constructed a deck of playing cards, shuffled them, and then dealt five card poker hands. Their program had to examine each hand to determine if it contained one of the standard poker hands of value, e.g. one pair, two pair, three of a kind, etc.

Our students find classic computer science topics engaging. All three of these projects involve classic computer science topics; greedy algorithms, depth first search and simulation. Two of the three projects involved a puzzle or game, but there were three other assignments involving puzzles or games that did not rank nearly as high. None of these projects involved graphics or GUIs. Other assignments that did involved graphics or GUIs did not rank as high. This leads us to believe that one key to increasing student engagement is an increased focus on applying the current course material and less on whether we think the students will find the application interesting.

5.2.5 The Most Successful Projects?

The two points in region E of Figure 1 have slightly lower Engagement means, but significantly lower Frustration means than the projects in the previous section. These two points represent early semester projects in CS1 and CS2. One of the points represents the second CS1 project, where the students used the ACM Graphics library to draw a snow scene. The project serves as a nice introduction to objects and methods. Students are encouraged to use their imaginations when designing their scenes and some are very elaborate. The other point represents an array review assignment given in CS2. Students represented a polynomial using a 1D array. They had to develop methods to evaluate it and to find roots using a binary search. Both of these projects were successful, but the lower Frustration means on these projects does not imply that they were better projects than those in the previous section. Both were designed as gentle introductions/reviews. In general, we would rather see more challenging projects such as those in the previous section where both the Engagement and Frustration levels are a bit higher.

6. LESSONS LEARNED

There are some repeated themes and lessons that have emerged over the course of this research.

1. Computer science is engaging. Claims that we need to find problem domains that are particularly interesting to our students are simply not borne out by the data. Students want their projects to apply what is being discussed in class, and will be engaged if that is the case.
2. Students enjoy a challenge, and challenges imply at least some frustration. Our most engaging projects all had higher frustration rates than some others. Finding the balance between challenge and frustration is difficult. It wasn't unusual to find Frustration scores for individual projects at both ends of the scale. This is primarily due to the diverse skill levels of students in the classes.
3. We should have the courage to recognize our bad assignments and either remove them entirely from use, or at least completely rework them before they are used again. Projects with very high student frustration reduce engagement and adversely affect student self confidence and interest.
4. Managing frustration and engagement should be done incrementally, on a project by project basis. The survey data often give us a clear direction on how to achieve it. We were frequently surprised by how consistent student feedback was. Undoubtedly this is in part due to students talking with each other in the lab, but that doesn't make the feedback less valid.
5. Students really appreciate the chance to give feedback on projects. Sometimes they felt a need to let go of frustration, but often they provided thoughtful insights into the projects. Several students queried us at the end of the semester as to how the surveys would be used and whether we would continue to collect this information in future semesters.

7. CONTINUING WORK

We plan to continue to use the same survey in our courses. With more data we will be able to show patterns of feedback within individual courses and show how we have improved specific programming projects based on student feedback.

8. REFERENCES

[1] ACM Curriculum Committee on Computer Science. Curriculum '68: Recommendations for the undergraduate program in computer science. Communications of the ACM, 11(3), March 1968, 151-197

[2] ACM Curriculum Committee on Computer Science. Curriculum '78: Recommendations for the undergraduate program in computer science. *Communications of the ACM*, 22(3), March 1979, 147-166.

[3] ACM Curriculum Committee on Computer Science. Computing Curricula 1991. *Communications of the. ACM* 34(6), Jun. 1991, 68-84.

[4] Denning, Peter J., et al., Computing Curricula 2001 Computer Science Volume, December 15, 2001, Available on-line at: http://www.sigcse.org/cc2001/index.html

[5] Dictionary.com, Available on-line at: http://dictionary.reference.com/

[6] Margolis, Jane and Fisher, Allen, *Unlocking the Clubhouse: Women in Computing*, MIT Press, Cambridge MA, 2002.

[7] McGettrick, Andrew, et al., Grand Challenges in Computing: Education – A Summary, *The Computer Journal*, Vol. 48, no. 1, The British Computer Society, 2005.

[8] Parlante, Nick, et al., Nifty Assignments, Available on-line at: http://nifty.stanford.edu