A Problem of Frustration and Demotivation in Computer Science Course 58

A Design Study Proposal

Situative Design Project Study Proposal

ED 333A

Kristine Hanson

Edmund Shen

Wuping Lu
Introduction

In America and the world at large, technology is transforming society. We have welcomed it into our lives and altered our daily habits as a result. This trend has no end in sight, rather it is on the increase. A cooperative study done by NPR, the Kaiser Family Foundation, and the Kennedy School found that “Americans under age 60 are rapidly adopting new technologies in their homes. More than half (52%) of those who have a computer at home bought their first computer in the past five years.” And this technological movement has spread to more places than the home. “Americans say computers are a necessity at work. More than two-thirds (68%) of working Americans use a computer at work, and 84% of them say it is essential for their jobs.” Whether at home, work, or play, technology is rampant now.

This technological boost requires support; it requires network administrators, database administrators, web-site developers, computer programmers and more. The continued need for the technologically expert and the current lack thereof is evident in most want-ads posted or any HR department. So where are these needed techies? How can our colleges, universities, communities generate more technologically trained students and employees? One solution to this problem might lie within a specific case found at South Western University.

Learning Environment

CS 58 is an undergraduate computer science course at SWU entitled Introduction to Computer Programming. The focus of the course is to teach students the “nuts and bolts” of programming using Java. The course website details out the objectives of the course saying 90% entails training students on basic programming constructs, e.g. types (arrays, strings, …), variables, control structures (sequential flow, loops, …), abstract data types, recursion, and I/O. The desire is for students at the end of the semester to independently be able to write a program of a few hundred lines of code without the help of a TA. In fact the course outline is specific enough to say that one goal is to “help students understand the rudiments of the environment and the resources available to help them program (editors, basic computer architecture, compilers, libraries, reference manuals, on line help), [and to] help them to become independent of "hand-holding by a TA or an instructor." The other 10% of the course objective deals with introducing students to key programming concepts, e.g. abstraction, correctness, efficiency, reuse, and translation.

The course is taught in lecture format during 50 minute classes every Monday, Wednesday, and Friday. Since the course is the first prerequisite to the Computer Science and Computer Engineering majors, and since many other students of other majors exhibiting interest in learning computer programming enroll as well, the class lectures are very large in size, 100-150 students. The professor lectures using power point and discusses the course subjects, often stepping through coding examples to explain the individual line’s meaning. There are few opportunities for questions with so large of a group. Questions and exploration are to be handled during lab time, in the designated
computer labs where students complete their projects (normally one every week or two) with the help of the TAs on duty.

**Learning Problem**

Many difficulties exist in teaching this subject matter, especially given the large class size, but the problem our study will focus on is that of audience. How can the professor and TAs deliver the objectives of the course to both experienced and novice programmers, while eliminating frustration to the novice programmers and keeping the experienced programmers challenged. As stated, this is the first course to the CS and CE majors, and as such it captures a wide range of students – those discovering programming for the first time to others who have coded since early childhood. We don’t want to neglect that some enrolled students are simply exploring programming for personal interest or due to potential interest in the CS or CE majors. This section of the class audience is smaller, but very integral to the stated learning problem.

**Learning Goal**

A member of our team having attended SWU and hearing many students – no longer CS or CE majors – complaining of the difficulty and frustrations experienced in CS 58, we as a team would like to study the causes of why many potential computer science students lose interest in the field of programming and discontinue that academic path after taking the course. We define potential to mean a student who would be attracted to the field of programming, but becomes disinterested by the course set up as opposed to the subject matter. The object of this study would be to understand how to improve the course design with the hopes of maintaining as many CS and CE majors as possible. Our goal is to sustain and increase excitement for the field of programming, thus enlarging the population of technologically educated students. The demand for these students in the workplace is great. With technology ever expanding into our society, these students will be valuable in supporting and creating the new technological advancements in the world.

**Learning Principle**

To analyze the learning problem given this case, we have chosen two principles discussed in James Greeno et. al publication entitled *Cognition and Learning*.

**Principle 1 - Environments of participation in social practices of inquiry and learning.**

*Learning environments can be organized to foster students’ learning to participate in practices of inquiry and learning and to support the development of students’ personal identities as capable and confident learners and knowers.*

The current design of CS 58 does not allow much student participation in inquiry or learning. We hypothesize that due to this paucity of socially engaged learning, many students lose motivation and interest in programming.

Classroom
In the classroom, the teacher dominates the lecture and assumes the role of imparting knowledge to the class. Students are left to formulate their individual understanding, with rare opportunities to ask questions. Their few allowed questions may or may not assist other students in their personal understanding. The tools utilized are simple power point presentations that allow for one-way instruction, teacher to student. This setting may create high frustration levels, impatience, and boredom for experienced programmers if the lecture is geared towards the novice programmer, or too much time is spent on answering the questions of simple programming basics. The skilled programmer would feel that they are not taking part in the conversation to impart their knowledge, nor are they developing or strengthening their own understanding. Likewise, if the lecture is directed towards advanced programmers, the beginners will feel helpless in their lack of understanding and contribution to the class. They will be overwhelmed with the information, unable to make sense of the teachings, and perhaps intimidated to inquire for clarification.

Regardless of the level of the student, this lecture system for CS 58 likely cultivates a sense of powerlessness, devalue, and disregard for the student and their possible contributions to the class. It seems to be a highly demotivating technique for all students involved.

Computer Lab

In the computer lab, the students are again left alone to solve the lab assignments on their own. TAs are on duty for most of the day, but are instructed to designate highest priority to pass-offs (A pass-off is when someone is done with their program and needs to show their work to the TA for a signature of approval.) Once the pass-offs are handled, they are free to assist those with questions, but charged to help the students help themselves – one of the stated course objectives.

Even more so than in lecture, students are left alone to grasp the expected programming knowledge. If outgoing, and a peer is willing, they may solicit help from a peer, but no structure is established to facilitate such interaction. Here is where the highest levels of aggravation and inferiority are bred. There is potentially even less opportunity for questions than in lecture, depending on the busy schedule of the TA. And without answers to questions, oftentimes the student can progress no further in their program. Their work completion and comprehension level remain stagnant until assistance is provided.

Principle 2 – Support for development of positive epistemic identities. Learning environments can be organized to support the development of students’ personal identities as capable and confident learners and knowers.

Currently in CS 58, feedback and public acknowledgment of the students and their capabilities to learn and know is minimal. This lack of recognition and individual support may create negative identities or non-existent identities for students.

Classroom

In lecture, the only real opportunity to create and develop an epistemic identity is through questions. This situation favors the experienced programmers. If they ask
advanced questions, or flaunt their knowledge through their comments, they may develop a positive identity as one with more experience and learning than the majority. The beginner, will ask questions only to clarify or gain a greater understanding. Their basic questions will display their lack of knowledge, rather than their capability to learn, and produce negative epistemic identities and embarrassment among their peers. Clearly the negative identity will serve to disinterest the student in the class and likely programming.

Computer Lab

Because the lab is so individualized, and pass-offs are privately done between the TA and the student, there is not much opportunity for social epistemic identities to be fostered at all. However there are chances for development of private identities. If the student is given helpful instruction by the TA or a peer, and then led to complete their project, or if they complete the assignment on their own, learning the expected concepts as they work, this will foster a positive identity within themselves. By nature, programming welcomes a sense of accomplishment. The creation process is very hands on, and the results display visually almost immediately. Thus the student after completing the assignment can sense that they’ve built something, something important for the class, and in process has learned the required skill. The computer lab may be helpful for creating individual positive epistemic identities as learners.

Proposed study

The purpose of our study is to use a situative lens in evaluating the learning environments in CS 58 that may cause students frustration or loss of interest in learning programming; therefore, causing them to discontinue pursuing an academic path in programming. To achieve this objective, we will focus on if learning activities are created and organized to foster active participation for students of all levels, and if the learning environment provides support for development of positive epistemic identities. Our qualitative research method will include the following:

I. Participants:
   College students at SWU enrolled in CS 58
II. Location:
   South Western University
III. Timeframe
   One semester

Methodology

Pre-Study Survey

The purpose of the pre-study survey is to determine the skill level of the student, their confidence in learning or in what they already know, and their motivation and interest level in programming. This survey will be conducted as the first assignment. The user will create a username and password to answer the survey. They will use that same username to answer weekly status checks and the post-study survey. This procedure will allow us to link the information on an individual basis to see progression within students. The pre-study survey will focus on these questions:
1) What programming class(es) they took before this class?
2) If they participated in any activity related to programming in whatever setting (e.g., high school extracurricular interest group, the community they are living, and online programming community) or not?
3) If they were taught programming on their own, by parents, and/or by their friends?
4) What motivated them to take this class?

**Online Status check:**
The online status check is a way of providing our study with more details and data on the student’s efforts, commitment level, time involved, as well as motivational shifts. Each week they will answer the same set of questions regarding:
1) How long did you spend on readings?
2) How long did you spend working on projects?
3) While in the lab or at home, did you work on your projects alone or in a group (1 or more)?
4) Did discussions you had with peers outside of the lab aid you in your project?
5) Did you attend all lectures?
6) Did you ever require assistance from the TA or a peer while in the lab? If so, who helped you?
7) If you were ever waiting for the TA’s assistance, how long did you wait? Was the TA helpful?
8) Rank your current interest level in programming

**Post-Study Survey:**
At the end of the one-semester study, a post-study survey will be conducted for all participants. The post study will focus on determining any changes in knowledge and interest in the domain. The pre and post study surveys will be linked so that we can analyze the changes by individual. The post study questions will concern:
1) What grade did they receive in the class?
2) How interested they are in continuing learning how to program?
   - Disinterested
   - Neutral
   - Somewhat
   - Strongly
3) Their current perceived skill level:
   - Beginner
   - Intermediate
   - Advanced
4) Do they still believe anyone can study hard to be good at programming even if he or she do not have previous programming experience?

**Interviews:**
Based on the results we get from the pre-survey, we will select 25 students of varying skill levels and motivation levels to interview once a week throughout the semester. In these interviews, the researchers will focus on the student’s self-assessment of his/her interaction with other people within the various learning activities they’re involved in and how these factors influence his/her learning of programming. How their motivation is changing, and what’s causing it. Do they feel they are learning? Is anything frustrating them? How do they feel they are contributing to the class – or do they?
Observations:
Observations will take place in the classroom and in the computer lab during TA hours. We will be sure to observe both empty, quiet times in the lab, as well as busy times such as nights when projects are due. We will be observing these key aspects:

1) If class activities and lab time provide and make use of “a rich variety of social and material resources for learning.” What tools are the students presented with to aid in their learning goals – computer applications, computer websites, professor office hours, TA hours and availability.

2) If class activities and assignments allow students “to contribute to socially organized learning activities, as well as to engage in concentrated individual efforts.” What social interactions are encouraged by the professor, TAs, or course environment. What social interactions spontaneously arise amongst the students. Does the TA address the computer lab students as a group, or as individuals. Does the environment – noise or bustle – in the computer lab make it difficult for students to work individually.

3) If class activities and assignments include “formulating and evaluating questions, problems, conjectures, arguments, explanations, and so forth, as aspects of the social practices of sense-making and learning.” How does the professor handle questions raised in lecture. How do the TAs respond to questions in lab. How do the TAs handle project pass-offs. What opportunities are students afforded to share ideas or learned concepts among peers or to the professor and/or TAs.

4) If class activities and assignments are arranged “in ways that complement and reinforce differences in patterns of social interaction and in expertise brought by students of differing cultural backgrounds.” How are beginner programmers’ questions treated by the professor and other listening students in lecture. How are the advanced programmer’s questions or comments treated by the professor and other listening students in lecture. Do advanced programmers assist the novices with their projects and questions in the computer lab?

Assessment:
Based on the data above, we are hoping to determine what aspects of the learning environment in CS 58 contribute to changes in motivation levels (for better or worse) and are effective in teaching the course objectives. Does the course design enable students to engage socially in learning and understanding? Within this course, what are the causes to students developing positive or negative epistemic identities? We will set up a comparative analysis combining data gathered in the pre-study surveys, status checks, post-study surveys, interviews, and observations.

Potential Findings
The findings we hypothesize to find at the end of the semester study are as following:
• The large lectures intimidate the novice programmers, discouraging them from asking questions to aid their understanding. As a result, they lose interest in lectures, and fall behind in comprehension. Their lack of contribution to lecture also fosters a sense of aloneness and a feeling of dispensability.

• Although the course objectives detail that to rely on a TA for help is disadvantageous and undesirable, the course structure pushes the beginners to seek TA assistance rather than help from peers. If the student always relies on the TA for help, they will never develop confidence as a learner, or feel that they have gained sufficient programming knowledge in accordance with the course expectations.

• Because of the demands on the TA by the large student load, the waiting time for TAs will cause impatience within the students. This will either serve to intensify frustration, or will draw the student to finding other means of help. If the students is resourceful, they may learn to access online sites as tools to complete their projects. This will probably widen the gap between novice and advanced. Novices will tend to wait on the TA and become frustrated. Advanced programmers will have more experience consulting online sources and succeed through their own means.

• Because the professor is faced with a student audience ranging from novice to skilled programmer, he will attempt to direct the lectures to the middle level. This will stretch the beginners beyond their comfort level and bore the knowledgeable programmers.

• The experienced classmates play a very important role for the novice students. Novice students and experienced students who interact in peer-to-peer social groupings will form positive identities and feel they are both contributors to the class. This sense of identity and purpose will help them maintain and/or heighten their interest in programming.

Proposed Solution

1) Class Arrangements
   a) Small Project Groups:
      i) Use Apprenticeship (pair up experienced with novice) approach to make small groups with 3-4 people for projects. Thus the experienced students can help the novice students to increase their motivation and confidence while lowering their frustrating level as well as gaining fluency in computer skills and increased interest.
      ii) Rotate the students to form different group creating peer-to peer learning environments with different people.
      iii) Rotate the student to play different rolls in the study group, such as the architect, designer, code writer, troubleshooter, report writer, and presenter, to make everyone fully participate as a team member. In this case, keeping the novice programmers motivated and the experienced programmers still challenged.
b) Friday Breakout Sections:
   i) Assign breakout sections on Fridays that meet in the computer lab and are taught by the TAs. The students will gain the advantage of smaller class sizes that will provide more time for questions in a less intimidating setting. It will also allow the student to learn in the context of what is being taught – in front of a computer. They will be able to ask project related questions and work with the tools (computers) that are necessary for project completion and understanding.

2) Assignments
   a) Projects
      i) Design projects with varying levels of complexity. Recommend suggested embellishments to the basic project criteria. Make these advanced elements optional, but offer extra credit on the exams for the added work. Display the fancy, in-depth projects in class for all to see and admire. This will provide the advanced programmers with added interest, without pressuring the novices.

3) Social Environment:
   a) Discussion Board
      i) Create a discussion board to allow students to share ideas about the projects and provide helpful links and tips for figuring things out.

Conclusion

A considerable number of students in CS 58 at South Western University experienced frustration and lost their interest in programming; therefore, causing them to discontinue pursuing an academic path in this field after taking CS 58. The purpose of this investigation is to evaluate the learning environments in CS 58 and how they impact students’ learning using a situative approach.

Our findings show that the learning environment lacks social practice and interaction of sense-making and learning, therefore inhibiting students’ active participation in learning. Learning activities are not organized to support the development of students’ personal identities as capable and confident learners and knowers, nor as contributors to the class. Findings may explain why students feel frustrated and discontinue an academic path in programming.

To create a learning environment which fosters active participation and development of positive epistemic identities, we recommend using group study, Friday breakout sections, projects with varying levels of difficulty, and a discussion board.
References

NPR, the Kaiser Family Foundation, and the Kennedy School. Retrieved October 9, 2005 from http://www.npr.org/ramfiles/atc/20000301.atc.06.ram
