Retention in the First Programming Course: A Context Based Approach

Krishnendu Ghosh, Miami University

Krishnendu Ghosh received his Ph.D. in Computer Science and Engineering in 2012 from University of Cincinnati and his masters degree in mathematics from University of Wisconsin-Milwaukee (2001). He is currently an Assistant Professor in the Computer and Information Technology Department at Miami University. His research interests include cyber security and formal verification.

Michele D Dickey, Miami University

Michele D. Dickey is a professor and program coordinator for the Instructional Design & Technology program in the Department of Educational Psychology at Miami University. Her research interests focus is on the intersection of technology and learning design.

Prof. Laurena Werner, Miami University

Laurena (a.k.a. Laurie) Werner is currently a retired and rehired Associate Professor in the Computer and Information Technology Department and an affiliate of the Computer Science and Software Engineering department at Miami University. She has been a member of the faculty since 1979. Her research interests are computer science pedagogy, and information assurance.

Retention in the First Programming Course:

A Context Based Approach

Introduction

Retention in introductory programming course (CS1) is challenging for a number of reasons and has been a major focus in computer science education research (Petersen, Andrew, et al, 2016). Students drop CS1 course mainly because they experience a disconnect with the expectations of taking a computing course and their motivations to take the course. Culture-related issues and under preparedness of students are also factors for students to drop CS1 course. Non-traditional students and non-computing majors cite lack of time and motivation as major reasons to drop CS1 (Kinnunen, P. & Malmi, L, 2006).

The CS1 course in this study is, Fundamentals of Programming and Problem Solving, an introductory three-credit hour programming course taught at the regional campuses of a mid-sized university (see Appendix A for course outcomes). The student population in the course is diverse in preparedness, motivation, and majors. The definition of diversity is not limited to the traditional definition of diversity in computer science education literature with respect to underrepresented groups. Student diversity in the course includes non-traditional students, students seeking an associate degree in IT or a bachelors degree in computing or non-computing majors. The course is required for majors in computer science and engineering and is an elective option for students in associate degree programs in information technology and other non-computing majors. Data collected from prior semesters show at least twenty percent of the students stop attending classes after the second midterm, held approximately four weeks before the end of the semester, and hence fail the course. The language used in the course is Java and procedural programming is taught. The course has two components: problem solving and programming using Java. The problem solving portion of the course focuses on development of pseudo code and the implementation portion focuses on writing programs in Java using the procedural approach. The prerequisite of the course is intermediate algebra. Returning students may have taken the math requirement several years before and often are in need of a review. Initial programming assignments are based on math problems and address data types, decision structures and loops.

In this paper, we describe an intervention to address the low retention rates in the course. Our focus is to engage students with a project that connects the students' interests with the concepts of the course. The goal of the intervention is to increase student retention. To that end, we seek to answer the following research questions:

(1) *How can we engage students in the first course in programming?*
(2) *What process or processes can be followed to engage students in the first course in programming?*

The intervention is a context based approach to a semester long project in the course. The motivation of the intervention is to connect the students with the concepts throughout the semester and to engage students to attend the class. We call the project a *context based project*. After the introduction of the intervention, data from several semesters have elucidated a higher

retention in the course and students have positive perception of the intervention. To the best of our knowledge, there has not been a *context based project* based on student's interest reported in the literature about CS1 course.

Background

Project-based learning and problem-based learning are instructional strategies that have been used in CS1 courses. Vega, et. al (2013) found that project based learning not only increases engagement among students, but incremental projects also foster student engagement. A scaffolding set of assignments form the project. Problem based learning in foundational computer science course was reported (Kay et al, 2000). Creation of project based incorporates features of problem based learning is important for student learning in CS1. Seeking problems that engage students is critical. Goldweber, et. al. (2013) found that socially relevant projects are intriguing for students. However, a majority of the student population may not be interested in pursuing the socially relevant projects in the course because of their individual and diverse backgrounds, experiences, majors and motivation. Hence, socially relevant problems can be limiting for most of the students to work as the topic of their project.

The use of context based computing was reported for engagement of non-majors in computing (Guzdial, 2009; Guzdial, 2010). DiSalvo et.al. (2011) argued that it is important for educators to find the reasons for engagement of the target learners rather than focusing on what motivates computer scientists. The context to increase student engagement is students' interests. A process model to introduce real world problems in the course has been reported (Rick, et al, 2012). Other interventions to increase engagement and comfort in CS1 found to be effective include the introduction of student mentors for support in the CS1 course. A student mentor is a student who took the same course in a preceding semester. Prior work (Newhall, T. et al, 2014), suggests that student mentors increase retention in computer programming courses. Success in student learning for a semester long project has been described (Bareiss, C., 1996). The objective of using a semester long context-based project is to increase student engagement in the first programming class in addition to structured assignments (worksheets or simple programs addressing a concept) that enhanced the learning of the programming concepts in the course.

The open-endedness of the *context based project* allows students to be creative in problem solving and work on a solution to a problem of their interest. In an open-ended assignment students have the flexibility to work on their own project and fulfill a given set of the requirements (Cliburn, D. et al, 2010). The open-endedness of the *context based project* connects the concepts of programming and forms a component of learning-by-doing to learn the utilities of programming to practice the utilities of programming (Layman, L. et al., 2007).

Design of Context Based Project

The *context based project* is comprised of a scaffolding set of assignments. After each assignment, the instructor provides feedback to students. The project is named as MyProject. The motivation to name the context based project is for the students to take ownership of their work. The set of assignments for the *context based project* are divided into two parts: the Prototype

phase and the Implementation Phase. The final assignment, MyProject-Process is based on the learning outcomes of the Prototype and Implementation phases.

Assignments	Assigned during	Assignment deadline
Prototype-0 (Prototype Phase)	Third week	Fifth week
Prototype-1 (Prototype Phase)	Sixth week	Seventh week
Prototype-2 (Prototype Phase)	Seventh week	Eight week
Software version 1.0 (Implementation Phase)	Eight week	Tenth week
Software Version 2.0 (Implementation Phase)	Tenth week	Twelfth week
MyProject-Process	Twelfth week	Thirteenth week

Table 1. Tentative schedule of assignments for MyProject in a 14-week semester.

The tentative schedule of assignments is shown in Table 1. There are total of five assignments that comprise the definition of the problem, development of pseudo code for the problem and implementation of pseudo code in Java.

Prototype Phase

The first part of the *context based project* is the Prototype phase. The Prototype phase focused on the pseudo-code of the project scoped from students' interests. The Prototype phase of the project was focused on pseudo code development for the project. The Prototype phase comprised of three assignments, Prototype-0, Prototype-1 and Prototype-2.

Prototype-0: The assignment is:

> Write a one page description based of your interests that you want to do in your project. Explain with an example what your project will accomplish.

This assignment is open ended such that students who are novices in programming are not overwhelmed by the idea of working on a semester long project. The assignment provided an opportunity to students to describe their interests. The assignment helped the students to connect project idea that they envisioned with their interests. The deadline for submission for this assignment was two weeks after it was assigned. The assignment was given to the students after data types and problem solving concepts were taught. The instructor provided feedback on the assignment with the aim of defining a scope of the problem that would be feasible for the semester. The context of the problem is an interest of each student.

Prototype-1:

> Write a one page description of the project based on the comments of the instructor stating the input and output, precisely.

Students defined and worked on the problem after receiving guidance from the feedback of the instructor. This assignment also allowed the students to connect the concepts of problem solving while defining the problem for the project. The instructor provided feedback with the aim of helping to shape the scope of the project into one that could be feasible for the semester.

Prototype-2:

Write the pseudo code for the project based on Prototype-1 and feedback by your instructor.
Specify the input, output and the steps that are required to get an output from a given input.
State three problem solving concepts that you have used in the pseudo code.
Use only English statements and nothing else.

The aim of Prototype-2 was to evaluate the precise description of the problem and algorithmic thinking. The description of the problem solving process in English statements is to guide in the implementation phase. Students use the English statements as comments in the code. By this time in the semester, students were taught decision structures, loops and introduction of arrays. This assignment is the pseudo code for the project.

Implementation Phase

The second part of the context based project is the Implementation phase. The Implementation phase focused on implementation of the project in the form of a Java program. The Implementation phase began after student submission of Prototype-2 and instructor feedback was provided for Prototype-2. In the Implementation phase, there are two assignments, Software version 1.0 and Software version 2.0.

Software version 1.0:

Write Java implementation of Prototype-2 and the feedback received. Use the steps shown in Prototype-2 as comments in your code. State five programming concepts/constructs that you have used in the Java implementation

This assignment is the implementation of the pseudo code in Prototype-2 in a Java program and feedback from the instructor. By this time of the semester, arrays and methods were already taught. After the submission of this assignment, students formed groups. After the submission of Software version 1.0, there was an in-class discussion amongst group members for the projects of each member in the group. At the end of the group discussion, students were asked to complete the following assignment. The assignment was used as a guide for discussions in the groups regarding the projects.

1. State two projects that were discussed in your group that you liked. Describe the projects as you learned from your group members.
2. State one project that you discussed in your group that you could have worked in a different way.

Please indicate if you strongly agree, agree, disagree, strongly disagree with, or are not sure with the following statements.

a. Students in my group came prepared and willing to participate.
b. Students in my group participated freely and were not intimidated by others.
c. Students in my group asked questions when in doubt.
d. My group summarized the procedural steps and important concepts in the assignment.
e. My group was satisfied with how our assignments turned out.

The aforementioned assignment helped the students to demonstrate their project based on their implementation and also, critique the projects of other students who were members of their group. The in-class assignment and reflection gave exposure to other student projects. Also, the

instructor provided feedback on the Software version 1.0 and additional features were recommended to be incorporated in Software version 2.0.

Software version 2.0:

> Modify the Software version 1.0 based on the feedback received. State clearly how you have incorporated the feedback in Software version 2.0.

The assignment, Software version 2.0. is a program that improved on Software version 1.0, based on the feedback of the instructor. Before the submission of this assignment, students learned input-output file handling. Enhancements from Software version 1.0 focused on the creation of methods and reading from input file or storing the results in an output file.

MyProject Process

After the end of implementation phase, this assignment was given to evaluate and reflect student learning during the prototype and implementation phases. The assignment is:

1. Reflect on the process of creating a prototype and implementation in MyProject .The project lifecycle consists of MyProject-Prototype 0, MyProject-Prototype 1, MyProject-Prototype 2, MyProject Software version 1.0 and MyProject Software version 2.0. Write two paragraphs.
2. If you were given more time (say another semester), state two features you could have added.
3. Which phase did you find difficult? Prototype phase or Software implementation phase? State a reason.
4. What are three things that you liked while executing the project?
5. What are three things that you think would improve the structure of the project?

MyProject Presentation.

On the last week of the semester, there was an extra credit assignment which was short presentation of five-seven minutes. The presentation addresses the following questions:

1. State your project briefly.
2. What did you do in the prototype phase? State the refinements.
3. Describe the implementation in Software Version 1.0. What are the challenges you faced?
4. What were your enhancements from the Software Version 1.0?
5. What did you learn while working on Prototype phase?
6. What did you learn while working on Software phase?
7. What were the learning outcomes of the course that were addressed in the series of MyProject assignments?

Feedback by the instructor

The feedback by the instructor after completion of each assignment in the *context based project* was critical. The scope of the project was determined by the instructor and communicated to the students in the feedback of the assignments. The set of assignments and the problem definition is

also dependent on the feedback of the instructor. The feedback created a dialogue between the students and the instructor of concepts during the execution of the project. Student learning of the concepts of problem solving, refinements in the process of prototype developments and implementation were evaluated.

Results

The assessment of the context based project was done using student surveys and qualitative methods such as reflections. Data from surveys were collected at the beginning and end of the project. Survey data was collected at the end of the Prototype phase and Implementation phase. The baseline of the study was the number of students in the course in Fall 2013 and Spring 2014 semester. Table 2 shows the data of the enrolment. After a last date of withdrawal, students can earn a grade. The number of students received a grade "W" after the drop date and before the end of the withdraw end date. The *context based project* was introduced in Fall 2014.
The data collected is from the sections taught by the same instructor across the semester. From the data available from Spring 2015, only one student failed per section. In Fall 2014, no students failed the class.
Data in the section 2 of Spring 2016 shows only one student received a failing grade, however, there were extenuating circumstances and the student had life issues (family obligations or illness) that impacted performance. In section (1) in Spring 2016, three students received failing grades. This section was held once a week in the evening and hence, it was challenging for returning students to cover the material if they missed two lectures. One student who failed reported life issues that impacted performance. In section (2) of Fall 2016, two students who failed had also had life issues.

	Fall 2013	Spring 2014(1)	Spring 2014(2)	Fall 2014	Spring 2015(1)	Spring 2015(2)	Spring 2016(1)	Spring 2016(2)	Fall 2016(1)	Fall 2016(2)
Total Enrolment	25	20	22	15	9	22	17	18	19	22
Number of students who withdrew	4	3	2	2	2	6	3	5	2	0
Number of students who failed.	5	4	6	0	1	1	3	1	2	2
Number of students who completed without a failing grade	16	13	14	13	6	13	11	12	15	20

Table 2. Student enrolment in the first course of programming.

Overall the attendance improved after the second midterm. Students were able to connect the concepts taught in class and were able to apply those the project.

Student surveys

Data from voluntary and anonymous student surveys were collected. A Likert scale was used in the data collection (strongly agree, agree, neutral, disagree and strongly disagree). Table 3 is the Pre-Post survey. The Pre survey was conducted before the start of the first assignment of the prototype phase. The post survey was conducted at the end of implementation phase.

	Questions
1	I am able to "Understand the programming concepts introduced in the class"?
2	I am able to “Write programs in Java based on the concepts introduced in the class”?
3	I am able to "“Find solution for a given problem”?
4	I am able to "Write a pseudo code for a given problem"?
5	I am able to "Write a program from a given pseudo code"?

Table 3. Survey questions for Pre-Post Surveys for MyProject.

Data showed more than 90% of the students surveyed agreed or strongly agreed with the questions. Additionally, there were two questions for the post survey for MyProject

1. The availability of a student mentor for the project helpful in the project.
2. I sought the help from the student mentor.

None of the students sought help from the student mentor explicitly for the project. There were students who sought help from the tutors but not for the project.

Table 4 and Table 5 are the surveys conducted at the end of the prototype phase and implementation phase, respectively.

	Questions
1.	I was able to make progress based on the feedback of the instructor during the prototype0, prototype-1 and prototype-2
2.	I was engaged in constructing prototype0, prototype-1 and prototype-3?
3.	I am able to " Write a pseudo code of a given problem "?

Table 4 Survey questions after the Prototype phase.

Data showed more than 70% agreed or strongly agreed for the first question in Table 4. The possible reason could be the project being open ended. Questions (2) and (3), more than 80% students agreed or strongly agreed.

	Questions
1.	I was able to make progress based on the feedback of the instructor during the software versions?
2.	I was engaged in constructing the software versions of the prototype?
3.	I am able to "Implement the pseudo code in a java program”?

Table 5. Survey questions after Implementation phase.

Data showed more than 80% of the students agreed or strongly agreed. It seemed students were engaged in the programming and had a perception they were working on their own project that they designed.

Qualitative Assessment

Overall the comments were positive from the students. A sample comments that are paraphrased from the MyProject Process assignment are below.

1. A non-computing major stated that he liked the course and enjoyed working on the project. At the beginning of the semester he did not think he could create programs based on his own ideas.

2. A non-traditional female student stated that the project helped her to decompose the problem into pieces and steps. She found the instructor watch the progress through the project and the comments provided by the instructor helped her to better understand the code. She felt that the project gave her real life example of creating a program.
3. A computing major student stated that the process of creating multiple prototype was redundant and was excessive. He had a good idea what he wanted to do and he felt the process was unnecessary lengthy.

The student reflections support that computing majors and student who have background in programming want to dive into programming at the earliest.

Conclusion

The *context based project* is a way to increase engagement of students in the first course in programming. The correlation with positive student perception at the end of the *context based projec*t and the decreased number of students failing the course, provide insights of student engagement in the course. The number of students failing the course decreased in comparison to the baseline data. The scaffolding set of assignments provides insight to answer the research question of processes that can be engaging for the student. Assignments based on pair programming (Nagappan et al., 2003) can be incorporated in the set of scaffolding assignments for a better learning experience. Meeting of students and student mentors could be mandatory for a stronger support during the implementation of the project. Student efficacy with respect to the project and background will be further investigated. In order to increase ownership of the *context based project*, MyProject will be renamed with the first name of the student as the prefix to Project. Further evaluation and adaptation of the *context based project* to the student population in the first course of programming could be useful to foster higher student retention rates in introductory computer programming courses.

References:

DiSalvo, B., & Bruckman, A. (2011). From interests to values. *Communications of the ACM*, 54(8), 27-29.

Newhall, T., Meeden, L., Danner, A., Soni, A., Ruiz, F., & Wicentowski, R. (2014, March). A support program for introductory CS courses that improves student performance and retains students from underrepresented groups. *In Proceedings of the 45th ACM technical symposium on Computer science education* (pp. 433-438). ACM.

Goldweber, M., Barr, J., Clear, T., Davoli, R., Mann, S., Patitsas, E., & Portnoff, S. (2013). A framework for enhancing the social good in computing education: a values approach. *ACM Inroads*, 4(1), 58-7

Guzdial, M. (2009). Education Teaching computing to everyone.*Communications of the ACM*, 52(5), 31-33.

Guzdial, M. (2010). Does contextualized computing education help?. ACM Inroads, 1(4), 4-6.

Rick, D., Morisse, M., & Schirmer, I. (2012). Bringing contexts into the classroom: a design-based approach. In Proceedings of the 7th Workshop in Primary and Secondary Computing Education (pp. 105-115). ACM.

Vega, C., Jiménez, C., & Villalobos, J. (2013). A scalable and incremental project-based learning approach for CS1/CS2 courses. *Education and Information Technologies*, *18*(2), 309-329.

Kay, J., Barg, M., Fekete, A., Greening, T., Hollands, O., Kingston, J. H., & Crawford, K. (2000). Problem-based learning for foundation computer science courses. *Computer Science Education*, *10*(2), 109-128.

Morrison, B. B., Dorn, B., & Guzdial, M. (2014, July). Measuring cognitive load in introductory CS: adaptation of an instrument. In *Proceedings of the tenth annual conference on International computing education research* (pp. 131-138). ACM.

Bareiss, C. C. (1996, March). A semester project for CS1. In *ACM SIGCSE Bulletin* (Vol. 28, No. 1, pp. 310-314). ACM

Kinnunen, P., & Malmi, L. (2006, September). Why students drop out CS1 course?. In *Proceedings of the second international workshop on Computing education research* (pp. 97-108). ACM.

Petersen, Andrew, et al. "Revisiting why students drop CS1." *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*. ACM, 2016.

Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., & Balik, S. (2003). Improving the CS1 experience with pair programming. *ACM SIGCSE Bulletin*, *35*(1), 359-362.

Cliburn, D. C., Miller, S. M., & Bowring, E. (2010). Student preferences between open-ended and structured game assignments in CS1. *In Frontiers in Education Conference (FIE), 2010 IEEE (pp. F2H-1). IEEE.*

Layman, L., Williams, L., & Slaten, K. (2007, March). Note to self: make assignments meaningful. In *ACM SIGCSE Bulletin* (Vol. 39, No. 1, pp. 459-463). ACM

Appendix A

Course Outcomes

1: To use a contemporary programming language and programming environment, and describe the language and environment using terminology appropriate for a technical audience

1.1: Describe the process of program translation from source code to intermediate or executable representation

1.2: Describe the concepts of syntax and semantics of a programming language

1.3: Describe and compare rules associated with declarations, including scope and lifetime, for program constructs such as variables, functions, and methods

1.4: Describe the data representation commonly used for integers, characters, and strings

1.5: Format and comment source code that adheres to a given set of formatting guidelines

1.6: Use command line tools to invoke the compiler and compiled executables

2: To be able to identify and eliminate errors in programs

2.1: Describe, using terminology appropriate for a technical audience, the difference between a syntax error, run-time error, and logic error

2.2: Read errors reported by the compiler and use those error messages to correct the syntax

2.3: Use techniques and tools for debugging programs

2.4: Design and document a complete set of test cases and use this to identify logic errors

2.5: Read and analyze code written by others, and identify errors in that code

3: To be able to specify, trace, and implement programs written in a contemporary programming language that solve a stated problem in a clean and robust fashion

3.1: Understand and appropriately apply primitive data types to represent information

3.2: Trace and use the common arithmetic operators within expressions that use parentheses and operator precedence

3.3: Describe, using terminology appropriate for a technical audience, trace, and implement programming control structures including pretest and posttest loops, counter-controlled loops, and conditionals

3.4: Understand and appropriately apply control structures, nested and un-nested

3.5: Use console and file input and output in a program

3.6: Apply one-dimensional and two-dimensional arrays to programming problems

3.7: Be able to use 3rd party class definitions, including those that represent strings, produce random numbers, perform math functions, format strings, perform console input and output, and ArrayLists.

4: To be able to solve programming problems using a procedural approach

4.1: Create and implement an algorithmic approach to a problem using functional decomposition

4.2: Determine necessary/desirable functions along with their needed structure (parameters, return types, etc.)

5: To be able to describe, trace, and implement basic algorithms

5.1: Describe, using terminology appropriate for a technical audience, trace, and implement linear search, non-recursive binary search, and at least one non-recursive sorting algorithm

5.2: Use standard library routines for searching and sorting arrays

5.3: Compare algorithms with respect to their efficiency, elegance, and readability

6: To be able to read, understand, and communicate technical information

6.1: Read, analyze, and summarize computer programming resources including textbook, API documentation, and help forums

6.2: Select and use examples, counterexamples, code alternatives, test cases, and diagrams to explain computer programming concepts