Design and Development of Self-Directed Learning (SDL) Modules for Foundations of Computer Programming Course

Dr. Gonca Altuger-Genc, State University of New York, Farmingdale

Dr. Gonca Altuger-Genc is an Assistant Professor at State University of New York - Farmingdale State College in the Mechanical Engineering Technology Department. She is serving as the K-12 STEM Outreach Research and Training Coordinator at Renewable Energy and Sustainability Center at Farmingdale State College. Her research interests are engineering education, self-directed lifelong learning, virtual laboratories, and decision-making framework development for design and manufacturing environments.

Dr. ilknur Aydin, Farmingdale State College

Ilknur Aydin is an Assistant Professor of Computer Systems at Farmingdale State College. Before coming to Farmingdale, Dr. Aydin was an Assistant Professor of Computer Science at SUNY Plattsburgh between 2009-2012. She received her Ph.D. and M.S. in Computer Science at University of Delaware. Her B.S. is from Computer Engineering at Marmara University in Istanbul, Turkey. Dr. Aydin's research is in the general area of wireless and mobile networks with a focus on transport layer protocols. In particular, Dr. Aydin has worked on the SCTP protocol, multihoming, congestion control issues, and network coding for the past several years. Dr. Aydin is the implementor of SCTP module in QualNet network simulator (a public software). Before coming to US for graduate studies, she had also worked as a software engineer in a tech company in Istanbul, Turkey on projects such as implementation of GPS based tracking software. Dr. Aydin is also interested in computer science education and increasing the recruitment and retention of women in computing

Design and Development of Self-Directed Learning (SDL) Modules for Foundations of Computer Programming Course

Abstract

In today's competitive world, the expectations from college graduates are much different than it was decades ago. The new graduates are now not only expected to know their area of expertise but also to educate themselves on the new and up-coming technologies. The rapid changes in technology along with the just-in-time and customer-centered project management made it a norm for engineers, engineering technologists, and computer programmers to have soft skills such as; self-directed learning, technical communication, and critical thinking in addition to their discipline specific knowledge and technical skills. Many students don't necessarily get exposed to self-directed learning skills in a traditional classroom setting. This makes it challenging for students to become self-directed learners after graduation. This study provides an overview of the design and development of several self-directed learning modules, along with the implementation procedure and analysis of the preliminary results.

Introduction

With the current pace the technological advancements are achieved today, it became necessary for recent college graduates to keep themselves up-to-date with all the innovations and technological advancements.[1, 2] Especially for engineering, engineering technology, and computer science majors, self-improvement and continuous learning after graduation became an expectation. The concept of continuous learning can present a challenge for many recent graduates as it may require self-directed learning. The traditional learning environment where the teacher presents the material does not provide an environment for students to gain self-directed learning skills. In self-directed learning, the learner has control on what he/she will learn, the approach they want to employ to learn the material and assess their learning.[3-5] The challenge with self-directed learning is to teach students the self-directed learning skill set. When students are not taught this skill set or did not get a chance to practice this skill set, they feel discomfort as self-directed learners. Negative experiences along with certain level of discomfort due to self-directed learning may be outcome of low self-perceptions of content learning and lack of personal interest on the topic.[6] The task of providing students with the self-directed learning skill set fall on the educational institutions and faculty. In an effort to provide students with this much needed skill set; faculty from various institutions developed materials and employed various approaches in their courses to teach students the self-directed learning skills. In University of Sydney, to promote independent study and lifelong learning, a series of online courses are implemented into freshman science course curriculum.[7] Project-based learning is also being used commonly to involve students beyond traditional learning and requires the students to use self-direction.[8] In a Computer Aided Engineering (CAE) course, students were asked to study and learn the lab material on their own and were forced to think and figure out the assignments on their own.[9] A series of modules that promote self-directed learning have been designed to be implemented into a senior level mechanical engineering technology course.[10] As the popularity of using social networks among students increased, educators used the social networking sites such as Facebook to teach students self-directed learning.[11] As the importance and the application of the self-directed learning increase, the need to assess students' self-directed learning skills also increased. A review of formative assessment and good application principles of the self-directed learning was examined so that proper assessment techniques can be developed.[12]

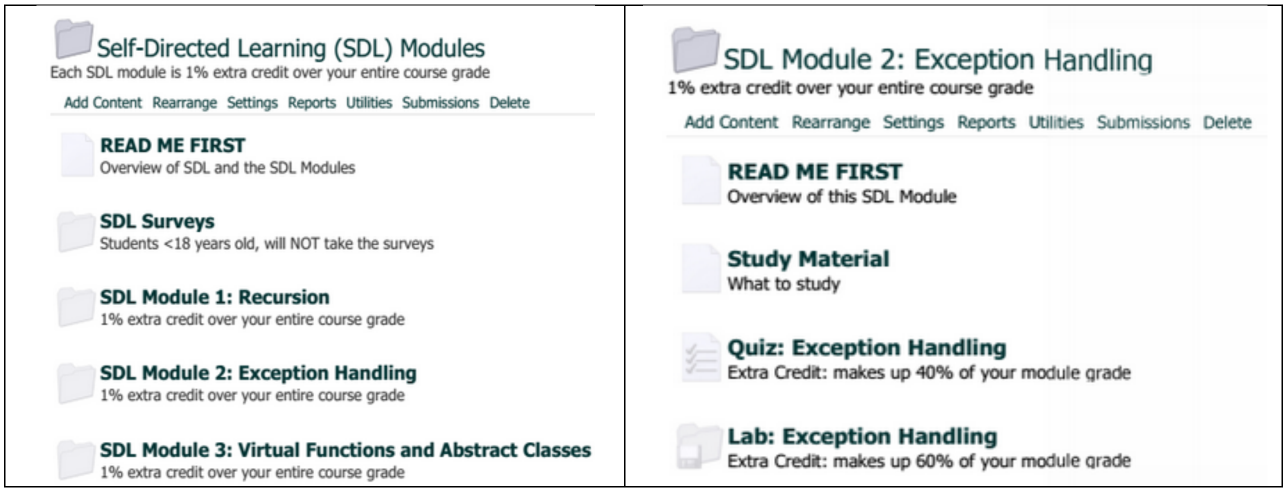
In an effort for students to gain the self-directed learning skills, which will last them lifelong, a variety of approaches have been implemented into the curriculum by faculty in different disciplines. These approaches include modules that can be implemented into the lecture material, projects that can be a part of senior design courses, and reading assignments that can complement the laboratory work. In engineering and engineering technology majors, the laboratory component in the courses, along with the senior design course component in the curriculum provide great opportunities to implement the self-directed learning component for students to gain the self-directed learning skill. In Computer Programming major, the inclusion of self-directed learning modules is also becoming very important. In order for the students to be competitive in their field and gain the self-directed learning skills, a series of self-directed learning modules are developed to be implemented in a sophomore level Foundations of Computer Programming II course, called BCS 230, at a 4-year State College. This paper provides an overview of the development of the self-directed learning modules along with the implementation procedure and implementation timeline.

General Course Description

BCS 230 - Foundations of Computer Programming II course is a sophomore level, 3-credit core course that covers advanced topics such as: arrays, pointers, strings, classes, data abstraction, inheritance, composition, templates, and overloading.[13, 14] Students have to successfully complete *BCS 120* – Foundations of Computer Programming I course with a C or better grade to be able take the BCS 230 course. *BCS 120 – Foundations of Computer Programming I* course is a freshman level, 3-credit core course that introduces C++ programming language. In this introductory level course, students learn to develop algorithms using top-down stepwise refinement as well as introduction to the some of the object oriented programming concepts and C++ syntax and debugging techniques.

During the Fall 2014 semester, BCS 230 course is offered as an in-class course that uses the online course management system called Angel.[15] That is, the course instructor shares the learning materials, announcements, and the assignments on the course site in Angel. During the Fall 2014 semester, the self-directed learning component is introduced to BCS 230 course through inclusion of three self-directed learning modules. The modules cover the topics of

Recursion, Exception Handling, and Virtual Functions and Abstract Classes, respectively. Figure 1 shows how the self-directed learning modules and an example module (Module 2 – Exception Handling) are shared with students on the course site in Angel.


Self-Directed Learning (SDL) Modules
Each SDL module is 1% extra credit over your entire course grade
Add Content Rearrange Settings Reports Utilities Submissions Delete
READ ME FIRST
Overview of SDL and the SDL Modules
SDL Surveys
Students <18 years old, will NOT take the surveys
SDL Module 1: Recursion
1% extra credit over your entire course grade
SDL Module 2: Exception Handling
1% extra credit over your entire course grade
SDL Module 3: Virtual Functions and Abstract Classes
1% extra credit over your entire course grade

SDL Module 2: Exception Handling
1% extra credit over your entire course grade
Add Content Rearrange Settings Reports Utilities Submissions Delete
READ ME FIRST
Overview of this SDL Module
Study Material
What to study
Quiz: Exception Handling
Extra Credit: makes up 40% of your module grade
Lab: Exception Handling
Extra Credit: makes up 60% of your module grade

Figure 1. Self-directed learning modules on Angel course site

Development of the Self-Directed Learning Modules

The pilot implementation of the self-directed learning modules is carried out during the Fall 2014 semester. Three self-directed learning modules have been developed. Table 1 shows the names and extra credit grade percentages of the modules. The details of each module are shared below.

Table 1. Self-Directed Learning Modules and Extra Credit Percentages

Module No	Module Name	Extra Credit Percentage
1	Recursion	1%
2	Exception Handling	1%
3	Virtual Functions and Abstract Classes	1%

Module 1 – Recursion

In computing, there are two major programming techniques for solving problems: iteration and recursion. In iteration, a problem is solved by repeatedly applying the same procedure until a solution is achieved. In recursion, a problem is solved by repeatedly reducing the problem into smaller instances of the same problem and tackling each smaller problem. BCS 120 and BCS 230 courses only teach the iteration technique, not the recursion technique. Only the students majoring in the programming track in Computer Systems department will learn the recursion technique later in the curriculum in *BCS 370 – Data Structures* class. Recursion is a very important concept and may not be easy-to-learn topic for some students; therefore we believe that there are benefits to introducing this topic earlier. Students who are not majoring in programming track will not learn about the recursion concept in another course. It is beneficial for those students to gain knowledge on the recursion.

In this module students will learn the following concepts related to recursion:

- What are recursion, recursive function, and recursive algorithm?
- What are direct recursion, indirect recursion, and infinite recursion?
- How to solve a problem using a recursive algorithm and how to write a program using recursion?
- Recursion vs. Iteration: When to use which one?

Module 2 – Exception Handling

Exception is an anomaly, a fault, or an unwanted event that happens when a program is executing. These unwanted events can be such as division by zero, opening a non-existent input file, or the user entering a text data while the program is expecting a number. Exception causes the programs to terminate abruptly; instead, we want the program to terminate gracefully by giving an error message to the user or recover from the exception and continue executing. This can be achieved by including exception handling code in the program.

Exception handling is an advanced concept for modern programming languages such as C++. BCS 230 class does not cover the exception handling concept, only the upper level Computer Systems courses such as *BCS 345 – Java Programming* and *BCS 370 – Data Structures* courses may cover this concept. Students who learn about the exception handling concept will benefit greatly in the upper level Computer Systems courses.

In this module students will learn the following concepts related to Exception Handling:
- What is an exception? What are the events that cause an exception?
- How to write Exception Handling code, that is try/catch blocks, in C++?
- How to throw an exception?
- Learning about and using C++ Exception Handling Classes
- How to create your own Exception Classes?
- What are the three Exception Handling Techniques: Terminating the program, Fixing the Error and Continuing, Logging the Error and Continuing?

Module 3 – Virtual Functions and Abstract Classes

Virtual functions make it possible to design a base class in an abstract way and force the derived classes to custom-implement the base class' methods according to their needs. This way classes don't need to be derived from scratch. Virtual Functions and Abstract classes are part of one of the three main principles of Object Oriented Programming – that is polymorphism.

Students in the Computer Systems major learn about the virtual functions and abstract classes' concepts in *BCS 345 – Java Programming* course, a required course for all students in the department. Introducing some of the concepts of in BCS 230 course earlier will greatly improve students' learning experience later in courses such as BCS 345.

In this module students will learn the following concepts related to Virtual Functions and Abstract Classes:

- What is a virtual function? Why and when do we need it?
- What is compile-time binding (static binding) vs. run-time binding (dynamic binding)?
- Virtual Functions with formal reference parameters and pointer parameters vs. value parameters
- What is a pure virtual function, and how to implement one?
- What is an abstract class and how to implement one?

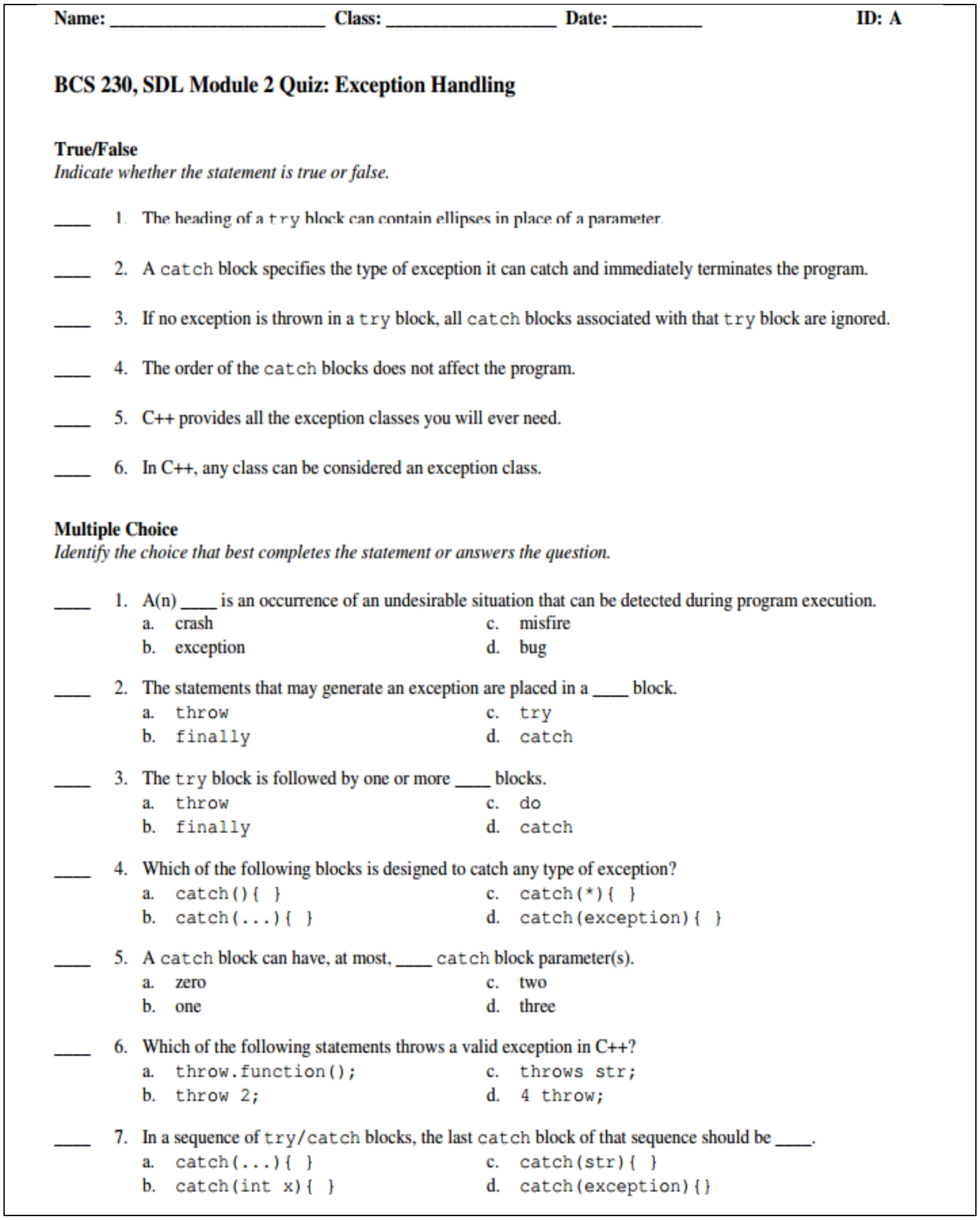
Pilot Implementation Overview

The pilot implementation of the self-directed learning modules was carried out during the Fall 2014 semester. Students were given the three modules in order of Module 1, Module 2, and Module 3. Students had 3 weeks in total to complete three modules. There were no restrictions on how many modules a student can complete. A student could pick and choose to complete one, two or all three of the modules within the given deadline. In order for the modules to be an effective representation of the self-directed learning concept, they were voluntary and extra credit. That is, students who choose not to participate in self-directed learning weren't penalized. In addition, students who wish to gain extra credit but who do not wish to complete the self-directed learning modules were offered 3 other assignments to complete for 1% extra credit each. These other extra credit assignments were a quiz and/or a programming assignment from the topics already covered in the course by the professor.

Throughout the self-directed learning experience, students completed the following 3-steps for each module:

Step 1: Students learn the topic on their own (*self-directed learning*). Course textbook[16] was the main reference to study the topic. The textbook has a dedicated chapter for each module. Additional resources such as tutorials, YouTube videos are also shared to complement the learning process.

Step 2: In an effort to measure students' learning and provide them with an instant feedback, each self-directed learning module was followed by a quiz. The quiz was administered via the online course management system (Angel). Students were provided with True/False and Multiple Choice questions. Students could take the quiz as many times as they wanted within the deadline of the module, and the highest achieved grade was counted. The availability of multiple attempts was designed to reinforce the learning. The quiz counted towards 40% of the module's overall grade. A portion of Module 2 quiz is shown in Figure 2.


Name: ______________________ **Class:** _________________ **Date:** _________ **ID: A**

BCS 230, SDL Module 2 Quiz: Exception Handling

True/False
Indicate whether the statement is true or false.

____ 1. The heading of a `try` block can contain ellipses in place of a parameter.

____ 2. A `catch` block specifies the type of exception it can catch and immediately terminates the program.

____ 3. If no exception is thrown in a `try` block, all `catch` blocks associated with that `try` block are ignored.

____ 4. The order of the `catch` blocks does not affect the program.

____ 5. C++ provides all the exception classes you will ever need.

____ 6. In C++, any class can be considered an exception class.

Multiple Choice
Identify the choice that best completes the statement or answers the question.

____ 1. A(n) ____ is an occurrence of an undesirable situation that can be detected during program execution.
a. crash c. misfire
b. exception d. bug

____ 2. The statements that may generate an exception are placed in a ____ block.
a. `throw` c. `try`
b. `finally` d. `catch`

____ 3. The `try` block is followed by one or more ____ blocks.
a. `throw` c. `do`
b. `finally` d. `catch`

____ 4. Which of the following blocks is designed to catch any type of exception?
a. `catch(){ }` c. `catch(*){ }`
b. `catch(...){ }` d. `catch(exception){ }`

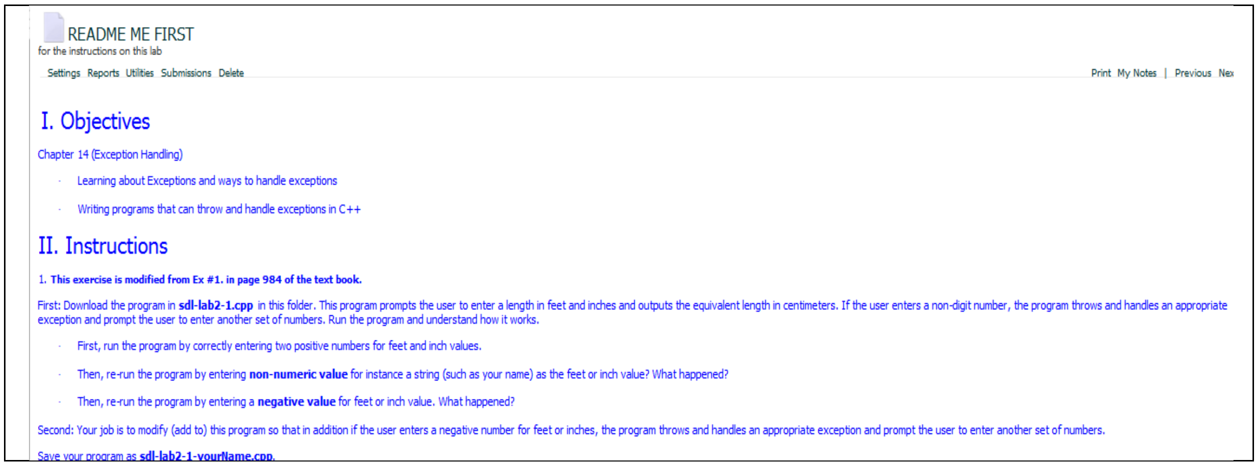
____ 5. A `catch` block can have, at most, ____ `catch` block parameter(s).
a. zero c. two
b. one d. three

____ 6. Which of the following statements throws a valid exception in C++?
a. `throw.function();` c. `throws str;`
b. `throw 2;` d. `4 throw;`

____ 7. In a sequence of `try/catch` blocks, the last `catch` block of that sequence should be ____.
a. `catch(...){ }` c. `catch(str){ }`
b. `catch(int x){ }` d. `catch(exception){}`

Figure 2. Quiz for Module 2 – Exception Handling

Step 3: In this last step, students were asked to complete 1-3 small programming assignments to apply what they learnt in that particular module. These programming assignments included: (i) modifying and/or extending a given program and (ii) writing a small-to-medium size program from scratch. The programming assignments counted towards 60% of the module's overall grade. A sample of Module 2 lab is shown in Figure 3.



README ME FIRST

for the instructions on this lab

Settings Reports Utilities Submissions Delete

Print My Notes | Previous Nex

I. Objectives

Chapter 14 (Exception Handling)

- Learning about Exceptions and ways to handle exceptions
- Writing programs that can throw and handle exceptions in C++

II. Instructions

1. **This exercise is modified from Ex #1. in page 984 of the text book.**

First: Download the program in **sdl-lab2-1.cpp** in this folder. This program prompts the user to enter a length in feet and inches and outputs the equivalent length in centimeters. If the user enters a non-digit number, the program throws and handles an appropriate exception and prompt the user to enter another set of numbers. Run the program and understand how it works.

- First, run the program by correctly entering two positive numbers for feet and inch values.
- Then, re-run the program by entering **non-numeric value** for instance a string (such as your name) as the feet or inch value? What happened?
- Then, re-run the program by entering a **negative value** for feet or inch value. What happened?

Second: Your job is to modify (add to) this program so that in addition if the user enters a negative number for feet or inches, the program throws and handles an appropriate exception and prompt the user to enter another set of numbers.

Save your program as **sdl-lab2-1-yourName.cpp.**

Figure 3. Lab for Module 2 – Exception Handling

The assessment process for the pilot implementation was carried out via pre and post-experience surveys. The surveys measured students' interest as well as their perception of the self-directed learning concept. Students' understanding of the self-directed learning material was measured both via the module assessments and module surveys. The implementation process and the online learning environment for the BCS 230 course are shown in Figure 4. The results of the pilot implementation's pre and post-experience surveys will be used to further develop the SDL modules to meet the needs of the students' learning experience.

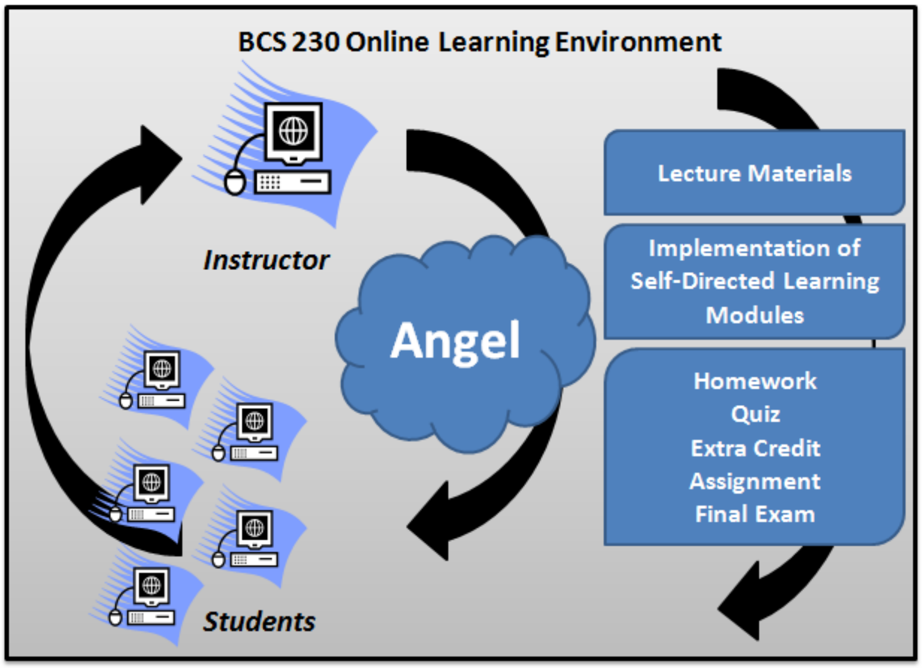


Figure 4. Implementation of SDL modules in the BCS 230 Online Learning Environment

Preliminary Results and Analysis

The pilot implementation of the SDL modules was completed during the Fall 2014 semester. Prior to the start of the pilot implementation, students were given a pre-experience survey. The goal of the pre-experience survey was to measure and understand students' interest in SDL. The pre-experience survey uses a 5-point Likert Scale. Upon completion of the SDL modules, students were provided with a post-experience survey. The post-experience survey measured students' feedback on the modules. During the Fall 2014 semester, BCS 230 course ran in two sections. Section-I had 16 students and section-II had 11 students; a total number of 27 students were enrolled in the course. Of the 27 students enrolled in the course, 4 students completed the surveys and the SDL modules. Although this is a low participation rate, it is also understood that participation to the SDL modules were 100% voluntary. It is also understood that the idea of self-directed learning may be an unknown and new concept to students, and that might have caused hesitation in participating to the SDL.

- *Pre-Experience Survey Question 1: I feel more engaged in courses that have interactive components such as a laboratory, hands-on project, use of software or simulation.*

  This question measures students' approach and attitude towards having interactive and active learning components as a part of their learning experience. This question's results are shown in Figure 5.

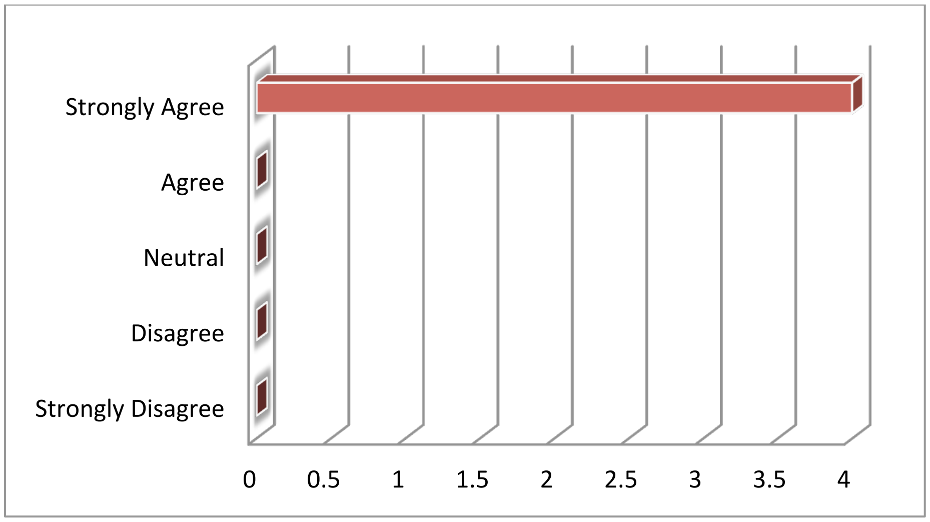


Figure 5. Pre-Experience Survey Question 1 Results

- *Pre-Experience Survey Question 2: Self-directed Learning is as effective as in-class or on-the-job training experiences such as internships or co/ops.*

  This question measures students' prior knowledge on the SDL concept, and whether it is as effective as internship or co/op experiences. Four students answered this question, and two students were neutral regarding to the statement. This is expected, as students may not have prior experience with SDL. This question's results are shown in Figure 6.

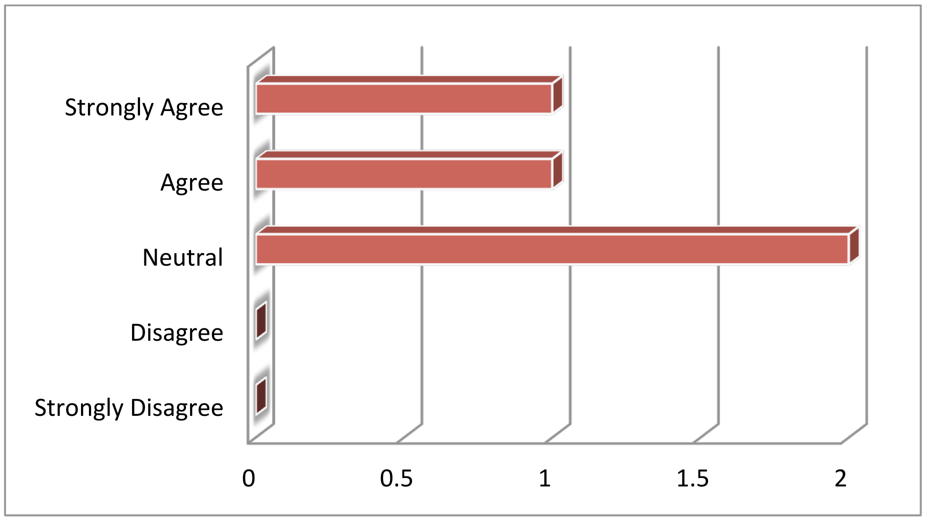


Figure 6. Pre-Experience Survey Question 2 Results

➢ <u>*Post-Experience Survey Question 1:*</u> *The Self Directed Learning modules helped me understand the concepts better.*

This question is designed to measure students' feedback on the effectiveness of the SDL modules. This question's results are shown in Figure 7.

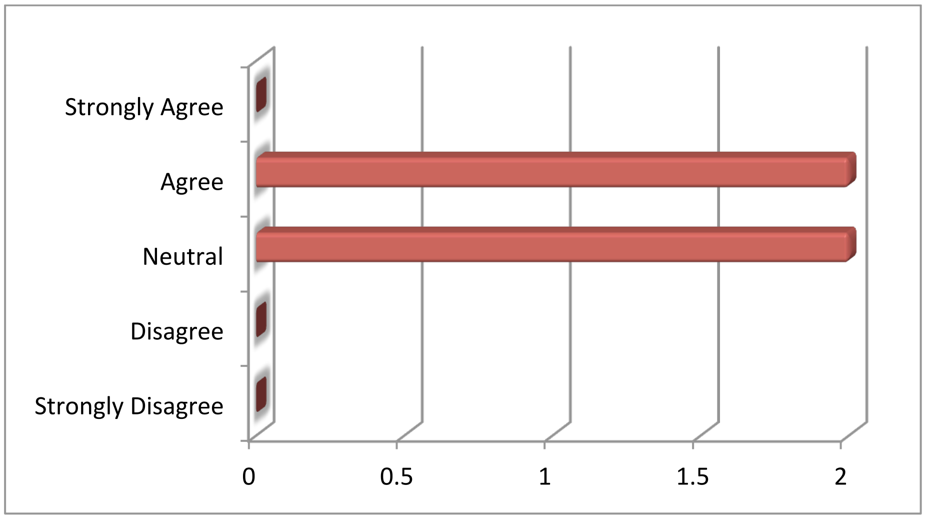


Figure 7. Post-Experience Survey Question 1 Results

- *Post-Experience Survey Question 2:* *The Self-Directed Learning modules were easy to understand and work-on.*

  This question is designed to measure user-friendliness of the SDL modules. This question's results are shown in Figure 8.

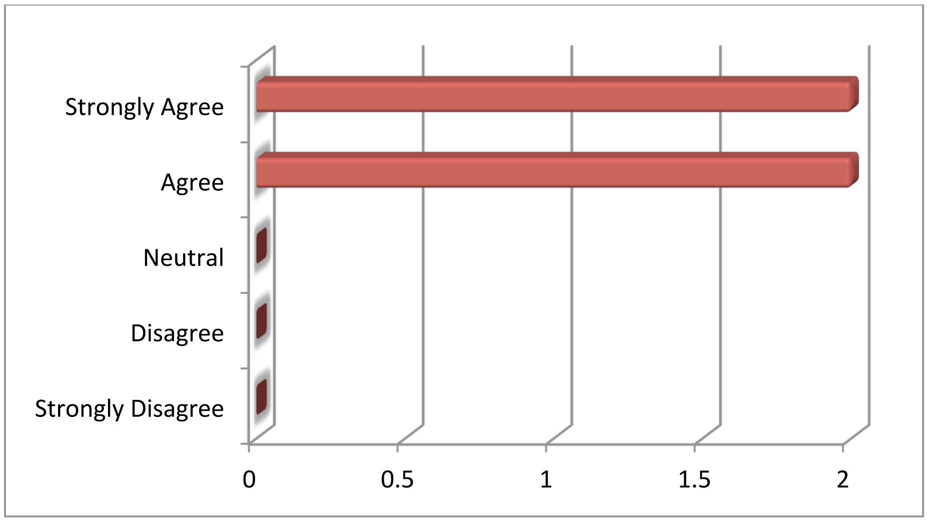


Figure 8. Post-Experience Survey Question 2 Results

- *Post-Experience Survey Question 3:* *I like the accessibility of the SDL modules online*

  This question is designed to measure if online platform is an effective method to deliver the SDL modules. This question's results are shown in Figure 9.

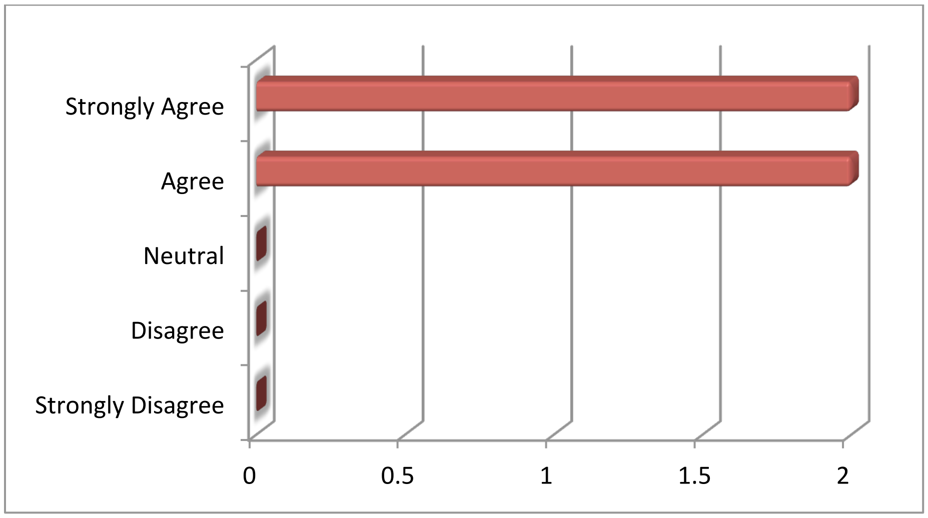


Figure 9. Post-Experience Survey Question 3 Results

The preliminary data shows that students who completed the SDL modules found the content of the SDL modules effective, easy to work with and SDL modules helped them understand the concepts better. Students also liked the accessibility of the modules through the online Angel platform. It is authors' goal to use the pilot implementation results and experience to develop additional SDL modules as well as to increase student participation.

Conclusions and Future Work

This paper provided an overview of the development of the self-directed learning modules and the analysis of the preliminary data of the pilot implementation of the SDL modules for *BCS 230 - Foundations of Programming II* course. The preliminary results will be used to develop additional SDL modules as well as to increase student participation.

The planned timeline for the implementation process of the self-directed learning modules are shown in Table 2. The data collection started in the Fall 2014 semester and will continue through the Spring 2015, Fall 2015, and Fall 2016 semesters.

Table 2. Two-Year Timeline to Implement SDL Modules in BCS 230 Course

Year	Semester	Activity
YEAR 1	Fall 2014	IRB approval for the pilot study, implementation of three SDL modules, collection of survey data.
	Spring 2015	Collection of performance data and survey data. Same SDL modules as Fall 2014 semester for comparison of student responses. Comparison of Fall 2014 outcomes with Spring 2015 outcomes.
YEAR 2	Fall 2015	Improvement on existing SDL modules and possible addition of new SDL modules and collection of performance data and survey data.
	Spring 2016	Collection of performance data and survey data, comparison for Fall 2015 data with Spring 2016 data and comparison of first year implementation outcomes with second year implementation outcomes.

The aggregated data over the four semesters will be analyzed to measure the student responses, interest in self-directed learning as well as to improve the existing modules and to develop new additional modules.

References

1. Altuger, G., and Chassapis, C., 2010, "Work in Progress – Preparing Students for Lifelong Learning in a Capstone Design Environment", Proceedings of the 2010 Frontiers in Education Conference, October 27-30, 2010, Washington DC
2. Altuger-Genc, G. and Chassapis, C., 2011, "Fostering Lifelong Learning In a Capstone Design Environment: An Implementation Assessment", Proceedings of the 2011 Frontiers in Education Conference, October 12-15, 2011, Rapid City,SD
3. Litzinger, T., Wise, J, Lee, S., and Bjorklund, S., 2003, "Assessing Readiness for Self-directed Learning", Proceedings of the 2003 American Society for Engineering Education Annual Conference and Exposition, June 22-25, 2003, Nashville, TN
4. Litzinger, T., Lee, S. H., and Wise, J., 2004, "Engineering Students' Readiness for Self-directed Learning", Proceedings of the 2004 American Society for Engineering Education Annual Conference and Exposition, June 20-23, 2004, Salt Lake City, UT
5. Litzinger, T., Wise, J.C., and Lee, S. H.,2005, "Self-directed Learning Readiness Among Engineering Undergraduate Students", Journal of Engineering Education, April 2005, pp: 215-221
6. Stolk, J., Somerville, M., Geddes, J., and Martello, R., "Work in Progress: Understanding Discomfort: Student Responses to Self-Direction", Proceedings of the 36th ASEE/IEEE Frontiers in Education Conference, October 28-31, 2006, San Diego, CA.
7. Peat, M., Taylor, C. E., and Franklin, S., 2005, "Re-engineering of undergraduate science curricula to emphasize development of lifelong learning skills", Innovations in Education and Teaching International, Vol.42, No.2, May 2005, pp: 135-146
8. Stewart, R.A, 2007, "Evaluating the self-directed learning readiness of engineering undergraduates: a necessary precursor to project-based learning", World Transactions on Engineering and Technology Education, Vol.6, No.1, 2007, pp: 59-62
9. Lasher, W.C., 2006, "Using a Course in Computer-Aided Engineering to Foster Life-Long Learning", Proceedings of the Frontiers in Education Conference, October 28-31, 2006, San Diego, CA.
10. Altuger-Genc, G., 2013, "Design and Development of a Self-directed Learning Component for a Mechanical Engineering Technology Course", Proceedings of ASEE Mid Atlantic Section Fall Conference, October 11-12, 2013, Washington, DC.
11. Altuger-Genc, G., 2012, "Self-directed Lifelong Learning Through Facebook: A Pilot Implementation Assessment", Proceedings of the American Society for Engineering Education Conference, June 10-13, 2012, San Antonio, TX
12. Nicol, D.J. and Macfarlane-Dick, D., 2006, "Formative Assessment and Self-Regulated Learning: A model and seven principles of good feedback practice", Studies in Higher Education, Vol.31(2), pp:198-218
13. Farmingdale State College BCS 120 course description - http://www.farmingdale.edu/academics/business/bcs/courses/bcs-120.shtml accessed on 01/01/2015
14. Farmingdale State College BCS 230 course description - http://www.farmingdale.edu/academics/business/bcs/courses/bcs-230.shtml accessed on 01/01/2015
15. Angel Learning Management System. http://www.blackboard.com/Platforms/Learn/ANGEL-Resources.aspx
16. Malik, D.S., "C++ Programming: From Problem Analysis to Program Design", 6th Ed. Course Technology, 2013, ISBN-13: 9781133626381.