Using Web-based Tools to Enhance Student Learning and Practice in Data Structures Course

Chao Chen

January 2014

1. Introduction

The purpose of this project is to enhance student learning and practice in the course ECE368 "Data Structures". The following tools are designed and adopted in the Fall 2013 semester: i) using web-based visualization tools to help students understand the construction and the dynamic updating of different data structures, and ii) updating programming projects to enhance students' practical skill of managing data efficiently. Assessment results show that the above tools helped students understand and apply the concepts of the data structures and associated algorithms in practical applications.

The rest of the report is structured as follows: The background of this project and proposed innovations are introduced in Section 2. The detailed description of the innovative tools adopted in Fall 2013 is described in Section 3. The assessment method and evaluation results are included in Section 4. Discussion of future work is summarized in Section 5.

2. Project Description

2.1 Background

ECE 368 "Data Structures" is a core course in the Computer Engineering program, required for all Computer Engineering majors. It is also a popular technical elective course for Electrical Engineering program. The class is scheduled for every Fall semester and meets twice a week for 75 minutes each session. I am the course coordinator and have taught ECE368 for seven semesters. Based on my past teaching experience, the current setting of this course has the following weaknesses:

1. Student background

This course requires the knowledge of object-oriented programming (OOP) and programming experience of C++. Although all students have taken the prerequisite course ENGR222/CS228 (Object Oriented Programming), their exposure to OOP and practice of C++ programming is limited since ENGR222/CS228 only lasts 5 weeks long and there is no associated lab session. In addition, there are no other programming courses between ENGR222/CS228 and ECE368 – For most students, that is a gap of one-to-three years. Thus, the students' programming background is generally weak, but varies depending on their individual experience and interest.

2. Project design

ECE368 is lecture-based, there is no lab session that students do not have in-class hands-on exercise and cannot get instant feedback from the instructor. Therefore, their practice of problem-solving skills using the concept of data structures is mostly limited to the programming

projects that are assigned throughout the semester. In the past, four medium-scale programming projects were assigned. In each project, students are required to write the whole program package, including the main function, functions of specific schemes, library or utility functions, readme files, and a Makefile to compile the files.

Although in each project description, the grading policy is clearly stated, actual grading is difficult as students may organize their programs in totally different manners and often the preferred modular design (or OOP model) is not followed. Therefore, the instructor needs to spend a lot of time debugging the codes submitted and trying to guess the students' ideas of solving the problems. In addition, if their original submitted codes did not work, the students lack the motivation to revise their programs, even with the instructor's feedback.

2.2 Proposed Innovations

My proposed innovation plan covers two areas: i) using web-based visualization tools for better illustration of different data structures and associated algorithms, and ii) employing the client-server prototype in programming projects. The first innovation helps the students better understand the theory of data structures; the second innovation helps them practice the skills of managing data structures efficiently in real applications.

1. Web-based visualization

Data Structures are an important way of organizing information in a computer. Many of the linear and non-linear data structures can be visualized. For example, a linked list can be displayed as a collection of boxes connected by arrows. If such structures are presented as web-based interactive visualization tool, students are able to see what data structure their code creates and how the data structure changes dynamically as they add/delete data and perform associated algorithms on the data structure. Such visualization tools would enhance their understanding of the theory of data structures.

Data Structures is a fundamental Computer Science and Computer Engineering course. Through thorough web search, I expect to find some existing online visualization tools that show the operation for primitive data structures. My plan is to collect a set of such demos related to ECE368, and to design homework problems that ask students to use such tools in getting the answers. Moreover, to tailor the use of these existing visualization tools to the coverage of our Data Structure course and my way of delivering the material, modification are expected as well.

2. Client-server prototype

I plan to use client-server prototype method in designing the programming projects. This method has two building blocks: *the prototype model* and *the client-server software development environment*.

The prototype model is a type of system development method. Following this method, a prototype is made first as an early approximation of the final product or software system. A prototype acts as a sample to test the process. From this sample, we learn and try to build a better final product. It is also a widely adopted method in software industry. Using the prototype model, an overall goal for each project can be designed along with multiple milestones in-between. Following a procedure like the industry software system design process, students will start with a

prototype according to preliminary user requirements, and then upgrade the prototype with user feedback and new requirements.

The client-server applications enable users and developers to input, process, store, and access data from anytime, anywhere, and any device. Providing such a client-server software development environment enables students to deposit their codes to the server side and debug/test through the client side. The client-server architecture will be web-based with user-friendly GUI (Graphical User Interface) at the client side. Therefore, students will be able to easily test their program with instant results. In addition, instructor can provide user's library or utility functions at the server side to students for the ease of development.

With the prototype model and the client-server software development environment, students will be required to follow the incremental programming procedure and the object-oriented programming model, which are embedded in the client-server prototype. This not only helps students to form good programming habit, but also helps the instructor in grading.

2.3 Benefits

Unlike Computer Science students, students major in Computer Engineering typically have weaker background and limited training in programming. However, lots of Computer Engineering graduates enter the professional fields as software engineers or test engineers. Data Structures teaches the fundamental techniques of organizing information efficiently. Therefore, enhancing the learning and practice in the Data Structure course is essential for Computer Engineering majors.

With the proposed web-based tools, students are expected to have a greater level of interest in learning the material and practice their programming skills. Moreover, by adopting the prototype method and client-server software development environment that are widely used in the software industry, students are able to improve their programming skills by forming good habits and to gain product development experience by following the procedure of real software systems. Both these practices will prepare them better to enter the targeted workforce.

3. Detailed Design

1. Web-based visualization

A set of online visualization tools (e.g., Java applets) have been collected to demo the operations for various data structures. Online questions for each demo have also been designed and implemented under the Blackboard learning system. Students must visit the given links, perform certain tasks for each data structure or algorithm, and answer a set of online questions (e.g., short-answer, fill-in-the-blank, multiple-choice, true/false). Those visualization tools also helped in teaching the material and delivering basic ideas in this course.

Table 1 summarizes the set of online animation tools and their associated topics. Each question set worth 10 points, the average of the 12 students for each question set is also given in the table. Through evaluating student performance on each question, the instructor can understand better how students digest the concepts as well as how these questions can be modified in future semesters.

Table 1. Online animation tools designed for ECE368 (Fall 2013)

number	topic covered	number of animations	number of questions	student average (out of 10)
1	linked list	1	1	10
2	stack	1	3	6.5
3	queue	1	1	10
4	recursive thinking	1	2	6.83
5	binary search tree	1	4	8.00
6	AVL tree	1	4	8.91
7	red-black tree	1	4	9.23
8	B-tree	1	3	9.00
9	hash table	1	3	9.45
10	sorting algorithms	7	4	8.05
11	graph traversal	2	6	7.27
12	topological sort	2	3	8.90
13	minimum spanning tree and shortest path algorithms	3	5	7.40

2. Client-server prototype

While designing the programming assignments, the instructor encountered some difficulties in implementing the client-server prototype method under a web-based project management system. The intended goal was to set up a client-server software environment so that at the client side, the instructor would write a web-based user-friendly graphical user interface (GUI), and the students could deposit their codes to the server side and debug/test through the client side. However, the instructor found that the students may having trouble understand and use this system without the knowledge of client-server model, GUI, and socket programming. Furthermore, not all the programming assignments can be easily implemented using the client-server model and a fancier GUI may not always be beneficial than the fundamental standard I/O. Even if some projects are implemented under this client-server model, the students would be required to decompose their code to fit the system structure, which may distract the original assessment goal of these projects.

Instead, a different approach was chosen that serves the similar purpose. For each assignment, I have written the prototype skeleton code. These are similar to the server structure where students can deposit their implementations. The test programs were also prepared. They serve for the purpose of the client software for the students to debug/test and display the results.

I have updated and added new programming assignments. Besides giving detailed description and instructions for each assignment, I have also written the prototype skeleton codes for the students to work with. This way the students would follow a guideline and focus on practicing the specific tasks intended for each assignment. For several assignments, test programs and primitive grading programs are prepared so that students can use to i) test their code

conveniently, ii) learn how to design various test cases to test the correct behavior of their software application.

Table 2 summarizes the programming assignments and their associated topics. Four of the assignments are modified from those from previous semesters; three are brand new. For each assignment, the detailed modifications are indicated as well. Each programming assignment worth 100 points, the average of the 12 students for each assignment is also given in the table.

Table 2. Programming assignments designed for ECE368 (Fall 2013)

number	topic covered	newly designed?	new enhancement	student average (out of 100)
1	statistician class (fundamental of C++)	Y	header files, skeleton code; test program	85.36
2	polynomial class (container class, dynamic memory)	Y	header files, skeleton code; test program	78.43
3	online book catalog (linked list)	N	header files, skeleton code	82.90
4	car wash center simulation (queue and discrete-event-simulation)	N	header files, skeleton code	79.50
5	huffman coding (tree)	N	header files, skeleton code	93.30
6	hash table	Y	header files, skeleton code; test program	81.90
7	sorting algorithms	N	header files, skeleton code; test program	94.60

3. Other Improvements

I have added an additional section of multiple choice and/or true/false questions in each homework assignment to enhance the students' understanding of basic theory of data structures. These questions are implemented under the Blackboard learning system. The course material including visual aids, course webpage, and reference links to online resources are updated.

4. Evaluation Results

In Fall 2013 semester, 14 students signed up for ECE368 in the beginning of the semester. Two withdrew in the early stage of the semester. In the first week, the students were asked to finish an online pre-survey under blackboard. This pre-survey has two parts: the first question asked them when they took the pre-requisite course (ENGR222/CS228), the second part consists of five questions testing their background on C++ basics (with two questions on C++ class, one question on function call, and two questions on class inheritance). Table 3 summarizes the students' background based on the pre-survey results (from the 12 students who stayed in the class).

Table 3. Pre-survey result (ECE368: Fall 2013)

questions	student response
When did you take CS228/ENGR222?	semester (number of students): Fall 2008 (1), Spring 2012 (3), Fall 2012 (6), Spring 2013 (2)
Questions on C++ basics (total: 5 questions, 50 points)	student average: 19.17 standard deviation: 9.00

It can be seen from Table 3 that the students took the pre-requisite in different semesters, with one of them took the C++ class five years ago. The results from the five questions on C++ basics show that the students are not very skilled in C++ programming (on average only 40% correct).

The benefits of the innovations were assessed from several aspects: instructor's observation, student feedback (including some quantitative measures). They are detailed in the following subsections.

4.1 Instructor's Observation

With the enhancements mentioned in Section 3, the instructor observed that:

i) It is easier delivering the basic ideas with the students going through the online demos.

ii) More students are turning in codes that are more readable and can compile without problem. The primitive grading programs helped in testing the codes automatically and pointing out possible errors in the code.

4.2 Student Feedback

1. Individual Project Assessment

Starting from mid-semester, I added an online survey for the programming assignment to see how the new enhancement would benefit the students. Table 4 summarizes the survey questions.

Table 4. Survey questions for programming assignments (ECE368: Fall 2013)

number	question
1	How much time approximately did you spend on this programming assignment? Please list total time in hours, and if possible, please break it down to time spent on the following tasks: • Planning (think before act) • Coding (the first version of your code) • Testing and debugging (including code revising)
2	Which part(s) of the project is(are) most challenging and took more time?
3	The primitive codes (.h and .cc files) helped me get started in planning and coding. (Please choose from strongly agree, agree, neither agree or disagree, disagree, strongly disagree)
4	Do you have any comments and suggestions, or any new ideas that can be added to the project?

It is found that the amount of time that an average student spent on a programming assignment as follows: project 4: about 9 hours; project 5: about 10 hours; project 6: about 10.3 hours, project 7: about 6.6 hours.

The student response to survey question number 3 for programming assignments 4-7 is depicted in Figure 1. It can be seen from Figure 1 that overall the skeleton codes helped students getting started in planning and coding. In addition, the neutral response for project 6 helped me re-evaluate the primitive code that I provided and update in future semesters.

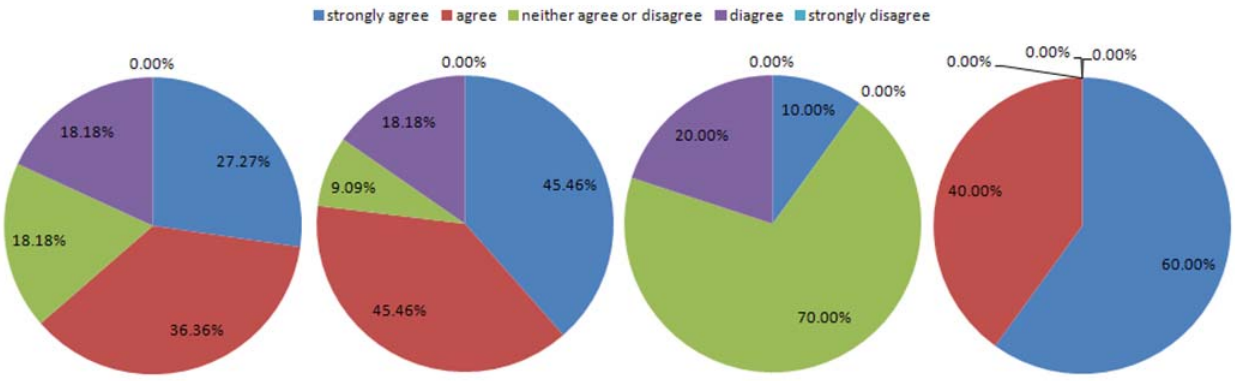

Figure 1. Survey results for question 3 (*The primitive codes (.h and .cc files) helped me get started in planning and coding.*) From left to right: student response to programming assignments 4 through 7.

Students also provided honest and valuable feedback on questions 2 and 4 (as listed in Table 4), which would be very useful for further improvement of the programming assignments in future semesters.

2. Course Overall Assessment

At the end of the semester, students were asked to finish an online post-survey under blackboard. This post-survey has two parts: the first part contains the same five questions in the pre-survey, testing C++ basics; the second part consists of 10 questions: the first 7 are opinion-scale based and the last 3 are open questions requesting for feedback and comments.

Table 5 shows students' performance on the basic C++ questions. The average improvement with respect to the pre-survey performance on the same set of questions is also included. It can be seen that the students' understanding of C++ is improved although the focus of ECE368 is not on C++ programming. Please note that two of the questions are focused on class inheritance, which is not practiced in programming assignments. Most of students answered incorrectly on at least one of these two questions in both the pre-survey and the post-survey. This is due to the fact that students only learned C++ in a short 5-week's class of ENGR222/CS228, there was not much practice on class inheritance. This can be a useful feedback to the instructor of ENGR222/CS228.

Table 5. Post-survey result: C++ basics (ECE368: Fall 2013)

questions	student response
Questions on C++ basics (total: 5 questions, 50 points)	student average: 29.09 standard deviation: 10.44 average improvement compared to pre-survey performance: (post-survey grade – pre-survey grade) / pre-survey grade = 95.45%

The other questions included in the post-survey are listed in Table 6 and Table 7. Students' feedback on the first 7 opinion scale based questions is summarized in Table 6 (average) and Figure 2 (distribution). It can be seen that all aspects of the course are rated high. The project assignments and the supporting documents for programming assignments were considered especially helpful in supporting the learning.

Table 6. Post-survey questions - part 1: opinion-scale based questions (ECE368: Fall 2013)

number	question	student response (average)
1	The **course content** was presented in class in a clear manner. Please choose from: strongly agree (5), agree (4), neither agree or disagree (3), disagree (2), strongly disagree (1)	4.09
2	How helpful were the **online animations and questions** in supporting your learning? Please choose from: significantly helpful (5), very helpful (4), somewhat helpful (3), only slightly helpful (2), not helpful at all (1)	3.18
3	How helpful were the **homework assignments** in supporting your learning? Please choose from: significantly helpful (5), very helpful (4), somewhat helpful (3), only slightly helpful (2), not helpful at all (1)	4.18
4	How helpful were the **programming assignments** in supporting your learning? Please choose from: significantly helpful (5), very helpful (4), somewhat helpful (3), only slightly helpful (2), not helpful at all (1)	4.18
5	In the programming assignments, how helpful were the supporting documents (**description files, program skeletons, testing files**) in your coding? Please choose from: significantly helpful (5), very helpful (4), somewhat helpful (3), only slightly helpful (2), not helpful at all (1)	4.09
6	How much **practical knowledge/skills** have you gained from this course? Please choose from: a great deal (5), a lot (4), a moderate amount (3), a little (2), none at all (1)	3.82
7	**Overall satisfaction** of this course Please choose from: excellent (5), good (4), average (3), fair(2), poor (1)	3.64

Table 7. Post-survey questions - part 2: feedback and comments (ECE368: Fall 2013)

number	question
8	If time allows, I wish the following topic(s) were covered in more detail: (Please choose from: C++ basics, complexity analysis; linear data structure (list, stack, queue) and applications, tree and applications, search and sort algorithms, graph and graph algorithms)
9	What was the most valuable part of this course?
10	It would have been more effective if the following can be done:

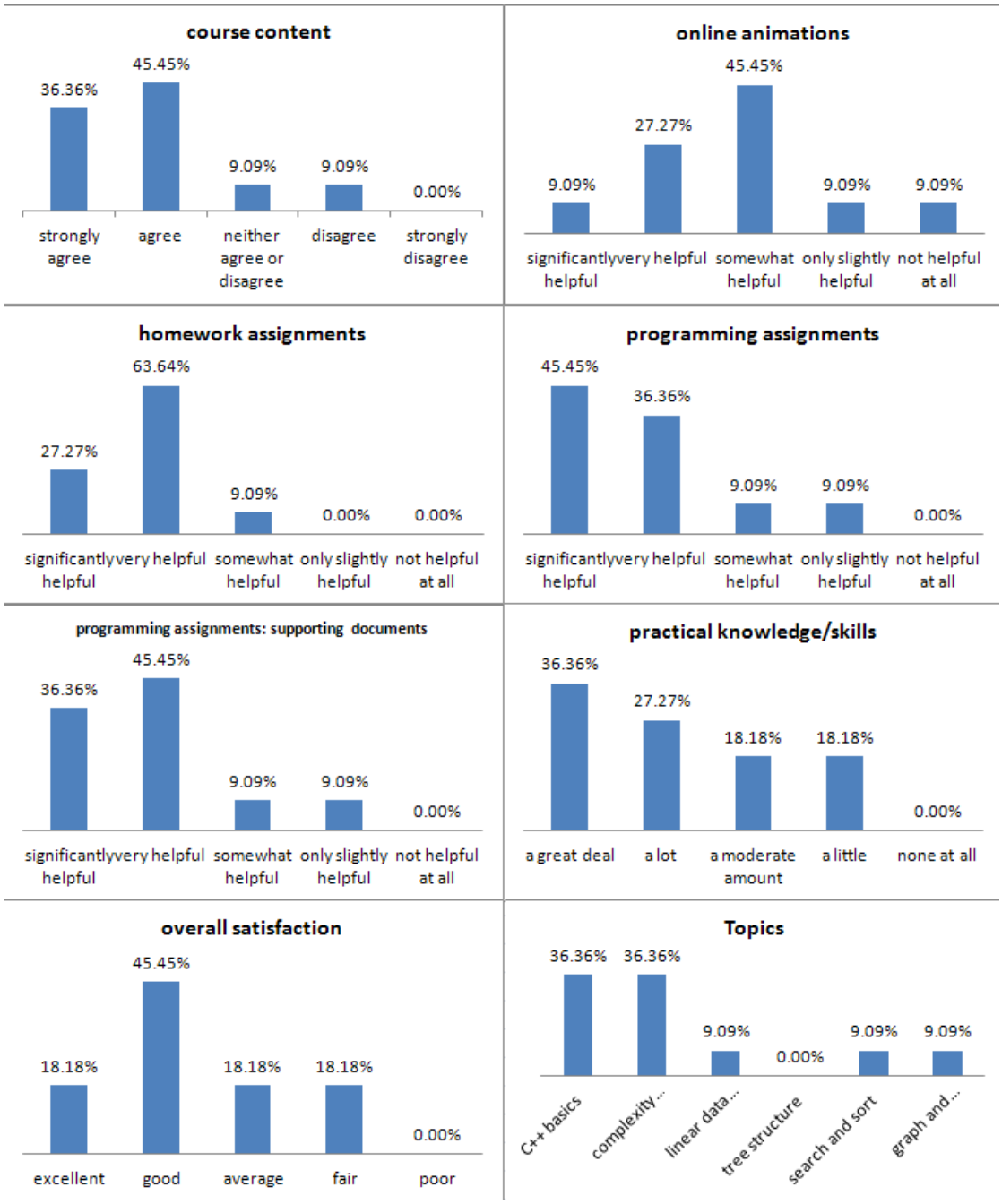

Figure 2. Post-survey results for questions 1-8.

Over 70% students wish that more time could be spent on the topics of C++ basics and complexity analysis. This is also repeatedly seen in their response to question 10. The most important reason is the inadequate preparation in the prerequisite course ENGR222/CS228, which only lasts for 5 weeks and students didn't have sufficient practice on C++ programming skills. The feedback has been forwarded to Computer Engineering Curriculum Committee for consideration. Students also provided other comments, which I will consider in future improvement of the course – detailed descriptions will be described in Section 5.

In the response to question 9, more than 60% students mentioned that practicing the programming skills through the projects was the most valuable part of this course. The following are cited from their comments.

> "I believe that the projects of the course were the most valuable piece. Although it took a while to really understand what was going on, after every project I could confidently say that I understood the process, the code, and the theory behind each. The projects were a very good mix of basics and new ideas that really helped to solidify the in-class lectures."
>
> "The depth and new knowledge provided about C++ data types and how to properly use them was the most valuable part of this course."
>
> I think the most valuable part of this course was the projects. Although they were tough and took a lot of time, I learned a lot about coding and what works and doesn't work just from sitting down and spending hours thinking about it."

5. Summary and Future Work

In Fall 2013, the following innovations were designed and deployed: i) web-based animation and visualization tools to help students understand the construction and the dynamic updating of different data structures, and ii) new and updated programming projects to enhance students' practical skill of managing data efficiently. Assessment results show that the above tools were effective and helped students understand and apply the concepts of the data structures and associated algorithms in practical applications.

Based on student feedback, the following improvement will be carried out in future semesters:

- In the beginning of the semester, upload online a brief introduction of C++ basics and have students finish a pre-class assessment test. The focus will be those topics closely related to the Data Structures class.
- Evaluate the current set of online animation tools and searching for better ones. In addition, update the online questions associated with each online animation tool, with reference to students' performance in Fall 2013.
- Update the programming assignments, with reference to students' feedback in Fall 2013. For example, add more detailed description and clarify certain functions that students had trouble with. Another new programming assignment on graph algorithms will be designed and included. This was not done because the students were given more time on each assignment. However, with better timing and well-prepared project package, the students can have chance to practice on the graph algorithms. In addition, the timing of the projects will be better aligned with corresponding topics in class.

- Try the client-server prototype method under a web-based project management system for one programming assignment. It could be a team project, where each member needs to implement part of the functions and upload to the server to merge with other parts that are done by other teammates.
- Continue gathering students' feedback on the online animation tools and programming assignments and use them for further improvement.