Innovative Redesign to Enhance Student Learning and Practice in Data Structures Course

Chao Chen

Indiana University – Purdue University Fort Wayne, Indiana
E-mail: chenc@ipfw.edu

Abstract

Unlike Computer Science students, Computer Engineering majors 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 course 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.

Throughout the past two years, innovative strategies have been adopted to improve the Data Structures course at Indiana University – Purdue University Fort Wayne in the following 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 helps the students better understand the theory of data structures; whereas the second helps them practice the skills of managing data structures efficiently in real applications. Both these practices will prepare them better to enter the targeted workforce.

This paper goes through the detailed design of the web-based visualization and client-server prototype based programming in the Data Structures course. Specifically, a set of online visualization tools have been collected to demonstrate the operations for various data structures. Online questions for each demo have also been designed and implemented. Students must visit the given links, perform certain tasks for each data structure or algorithm, and answer a set of online questions. Those visualization tools also help in teaching the material and delivering ideas during lecture time. In addition, new programming assignments are added or updated. With the prototype model and the client-server software development environment embedded in the design of such programming assignments, students are required to follow the incremental programming procedure and the object-oriented programming model.

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. Some suggestions for further improvement are also included.

The course redesign can serve as a first step towards offering this course in a hybrid mode. For example, the class can be implemented in flipped mode where the web-based visualization tools can help student learn course content at home; thus they have more time in class discussing and solving problems. The course also can be offered through distance learning for working adults who only come to campus for exams and problem-solving sessions.

1. Introduction

Unlike Computer Science students, Computer Engineering majors 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 course teaches the fundamental techniques of organizing information efficiently. Therefore, enhancing the learning and practice in the Data Structures course is essential for Computer Engineering majors.

Throughout the past two years, innovative strategies have been adopted to improve the Data Structures course at Indiana University – Purdue University Fort Wayne (IPFW) in the following 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 helps the students better understand the theory of data structures; whereas the second helps them practice the skills of managing data structures efficiently in real applications. Both these practices will prepare them better to enter the targeted workforce.

The rest of this paper is structured as follows: The background of the Data Structures course and proposed strategies are introduced in Section 2. The detailed description of the web-based visualization and client-server prototype based programming tools adopted is described in Section 3. The assessment method and evaluation results are included in Section 4. Assessment results show that the adopted tools were effective and helped students understand and apply the concepts of the data structures and associated algorithms in practical applications. Some suggestions for further improvement are included in Section 5.

2. Description

2.1 Background

ECE 368 “Data Structures” is a core course in the Computer Engineering program at IPFW, required for all Computer Engineering majors. It is also a popular technical elective course for Electrical Engineering majors. The class is scheduled for every Fall semester and meets twice a week for 75 minutes each session. Based on 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 a prerequisite course called “Object Oriented Programming”, their exposure to OOP and practice of C++ programming is limited since this course only lasts 5 weeks long and there is no associated lab session. In addition, there are no other programming courses between these two courses – For most students, this 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 thus students do not have in-class hands-on exercise and cannot get instant feedback from the instructor. Therefore, their problem-solving

practice 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 had to spend a lot of time debugging the codes submitted and trying to figure out the students' approaches 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 Methodology

Throughout the past two years, the Data Structures course have been redesigned with modifications mainly in the following 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 method helps the students better understand the theory of data structures; whereas the second method 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 existing online visualization tools that show the operation for various data structures. From them I will collect a set of demos related to ECE368, and design homework problems that ask students to run animations with self-chosen data input, and answer questions accordingly.

2. Client-server prototype

I choose to use the 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 [1,2]. 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, one 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 any client device [3]. 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 can be web-based with user-friendly graphical user interface (GUI) 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

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

3.1 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. 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

number	topic covered	number of animations	number of questions
1	linked list	1	1
2	stack	1	3
3	queue	1	1
4	recursive thinking	1	2
5	binary search tree	1	4
6	AVL tree	1	4
7	red-black tree	1	4
8	B-tree	1	3
9	hash table	1	3
10	sorting algorithms	7	4
11	graph traversal	2	6
12	topological sort	2	3
13	minimum spanning tree and shortest path algorithms	3	5

3.2 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 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, the prototype skeleton code is written by the instructor and provided to the students. 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.

With the prototype skeleton code, the students would follow a specific 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. For each assignment, the modifications are indicated as well.

Table 2. Programming assignments designed

number	topic covered	new enhancement
1	statistician class (fundamental of C++)	header files, skeleton code; test program
2	polynomial class (container class, dynamic memory)	header files, skeleton code; test program
3	online book catalog (linked list)	header files, skeleton code
4	car wash center simulation (queue and discrete-event-simulation)	header files, skeleton code
5	huffman coding (tree)	header files, skeleton code
6	hash table	header files, skeleton code; test program
7	sorting algorithms	header files, skeleton code; test program
8	graph shortest path algorithm	Header files, skeleton code; test program

4. Evaluation Results

In the Fall 2013 and Fall 2014 semesters, a total of 23 students (12 in Fall 2013 and 11 in Fall 2014) took ECE368. 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 inquires them when they took the pre-requisite course, the second part are questions testing their background on C++ basics. Table 3 summarizes the students' background based on the pre-survey results.

Table 3. Pre-survey result

questions	student response
When did you take the prerequisite course? (Gap till the current semester)	Percentage of students (out of total 23): within 1 year: 52.17% between 1 year and 2 years: 34.78% between 2 years and 3 years: 4.35% 4 years and beyond: 8.7%
Questions on C++ basics (Fall 2013: 5 questions)	student average: 38.34 standard deviation: 18.00
Questions on C++ basics (Fall 2014: 10 questions)	Student average: 57.27 Standard deviation: 24.94

It can be seen from Table 3 that the students took the pre-requisite in different semesters, thus their levels of familiarity to C++ programming vary. In Fall 2013, the second part of C++ background survey contains five questions: two on C++ classes, one on function call, and two on class inheritance. The results show that the students were poorly prepared in C++ programming (on average only 38.34% correct). In Fall 2014, the students were asked to watch a one-and-half-hour-long online C++ tutorial called "C++ Programming Tutorial Complete" [4] before working on the survey questions. In addition, the number of C++ basics questions is increased from 5 to 10, focusing on C++ basics, classes, and function call. The survey results show some improvement over the Fall 2013 data, with an average of 57.27%. Yet the results still show the lack of familiarity on C++ programming basics.

The benefits of the redesign efforts were assessed from several aspects: instructor's observation, student feedback (including quantitative measures) through the Blackboard eLearning system. 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

For each programming assignment, I added an online survey to see how the new enhancement would benefit the students. Table 4 summarizes the survey questions (Note that questions 4 and 5 are not for all projects).

Table 4. Survey questions for programming assignments

	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. (Choose from strongly agree, agree, neither agree or disagree, disagree, strongly disagree)
4	Did the test and/or exam programs help you find new bugs in your code?
5	The given test and/or exam programs helped me understand how to design test cases to verify code effectiveness and maximize fault detection. (Choose from strongly agree, agree, neither agree or disagree, disagree, strongly disagree)
6	Do you have any comments and suggestions, or any new ideas that can be added to the project?

It is found that the average amount of time that students spent on the programming assignments are as follows: 6.2 hours, 14.1 hours, 22.0 hours, 14.7 hours, 12.3 hours, 8.9 hours, 6.7 hours, and 7.4 hours, for project 1 through project 8, respectively.

The student response to survey question number 3 for the programming assignments is depicted in Figure 1. It can be seen that overall the skeleton codes helped students getting started in planning and coding.

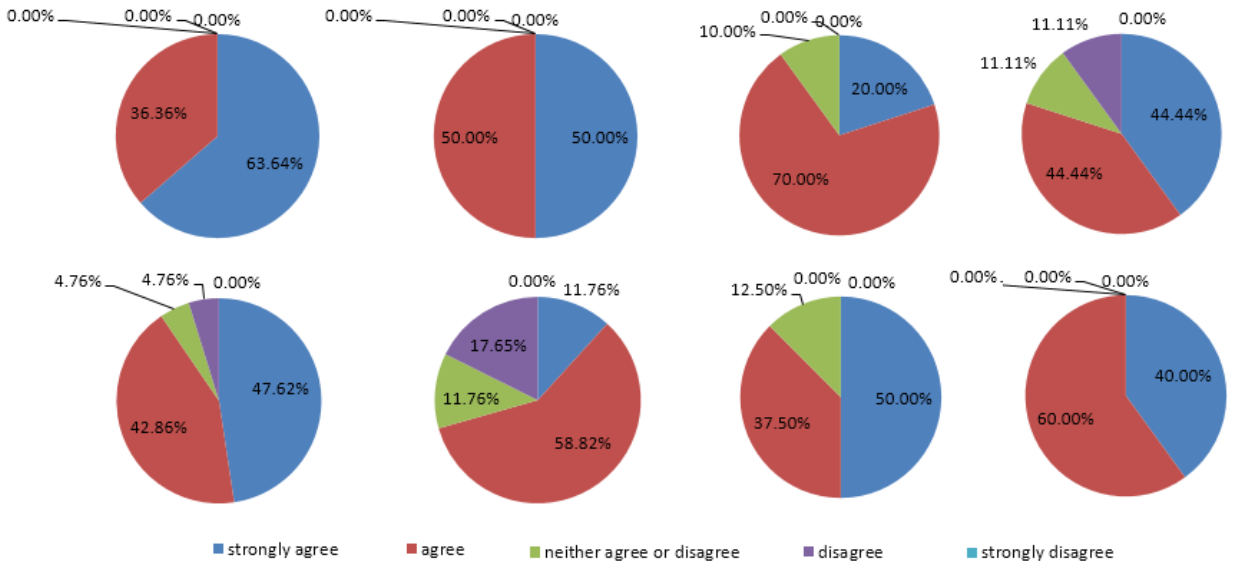

Figure 1. Survey results for question 3 (*The primitive codes (.h and .cc files) helped me get started in planning and coding.*) From left to right, then from top to down: student response to programming assignments 1 through 8.

Students also provided honest and valuable feedback on questions 2 and 6 (as listed in Table 4). The feedback is 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 consists 10 questions: the first 7 are opinion-scale based and the rest are open questions requesting for feedback and comments, as listed in Table 5 and Table 6.

Students' feedback on the first 7 opinion scale based questions is summarized in Table 5 (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 5. Post-survey questions - part 1: opinion-scale based questions

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)	4.24
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)	3.53
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)	4.29
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)	4.12
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)	4.19
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)	4.00
7	**Overall satisfaction** of this course Please choose from: excellent (5), good (4), average (3), fair(2), poor (1)	3.75

Table 6. Post-survey questions - part 2: feedback and comments

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:

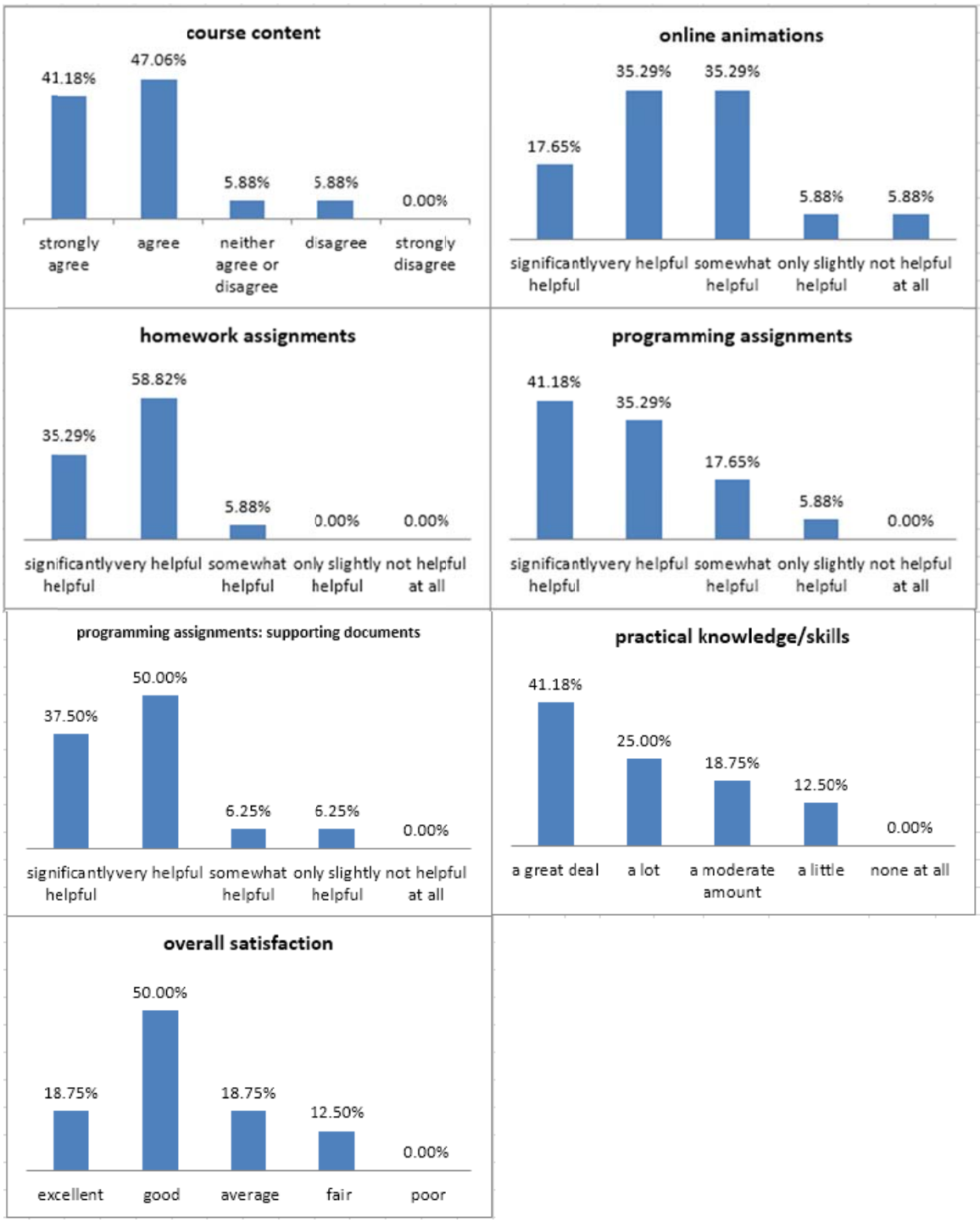

Figure 2. Post-survey results for questions 1-7.

Over 70% of the 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, which only lasts for 5 weeks and students didn't have sufficient practice on C++ programming skills. 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, close to 70% of the 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."
>
> "C++ reinforcement for future, and learning to store, modify, delete, and handle large amounts of data."

5. Summary and Future Work

In Fall 2013 and Fall 2014, the following strategies were designed and deployed in the Data Structures course: 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:

- Evaluate the current set of online animation tools and update if needed. In addition, update the questions associated with each online animation tool, with reference to students' performance in the past two years.

- Update the programming assignments, with reference to students' feedback in the past two years. For example, add more detailed description and clarify certain functions that students had trouble with.

- 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.

- Starting from Spring 2015 semester, a new 4-credit C/C++ course will be offered to replace the 5-week OOP prerequisite course along with its own 2-credit prerequisite on C programming. This new course has a moderate portion on C++ object-oriented

programming and has a 75-minute lab component per week. It is expected to greatly enhance students' understanding of C++ basics and their programming skills. Once all students have gone through the new prerequisite course, the coverage on C++ review in the Data Structures course can be significantly reduced. In addition, the first programming assignment, which is a simple practice on C++ classes, can be removed or replaced by another topic more related to the Data Structures course content.

In summary, innovative tools including web-based visualization and client-server prototype have been implemented and found effective. These redesign effort can serve as a first step towards offering this course in a hybrid mode. For example, the class can be implemented in flipped mode where the web-based visualization tools can help student learn course content at home; thus they have more time in class discussing and solving problems. The course also can be offered through distance learning for working adults who only come to campus for exams and problem-solving sessions.

Acknowledgment

This work was supported in part by the 2013 Summer Instructional Development Grant from the Center for Enhancement of Learning and Teaching at Indiana University – Purdue University Fort Wayne.

References

[1] Ian Sommerville, "Prototyping," in book "Software Engineering" (Section 2.3.1), 9th edition, Addison Wesley, 2011.
[2] "Software Prototyping", available at http://en.wikipedia.org/wiki/Software_prototyping
[3] Ian Sommerville, "Client-server architecture," in book "Software Engineering" (Section 6.3.3), 9th edition, Addison Wesley, 2011.
[4] "C++ Programming Tutorial Complete", available at http://www.youtube.com/watch?v=-WwGMNGRHdw