An Empirical Study of In-Class Laboratories on Student Learning of Linear Data Structures

Sarah Heckman
North Carolina State University
Raleigh, NC
sarah_heckman@ncsu.edu

ABSTRACT
Active learning increases student learning through collaborative engagement with materials during class time. A CS1.5 course at NC State, CSC216, uses active learning lectures involving short simplified think-pair-share in-class exercises to engage students with course materials. However, students still struggle with the course materials and several students do not successfully complete the course on their first attempt. To increase student learning and engagement, we conducted a quasi-experimental study incorporating in-class labs into two sections of CSC216 during the linear data structures unit in the Fall 2014 semester. Both sections completed in-class labs on the Java Collections Framework and iterators. One section completed in-class labs on array-based lists; the other section completed in-class labs on linked lists, in a counter-balanced study design. The active learning lecture delivery was used for the control section and an Exam was administered between the array-based list and linked list topics. Overall, we found no significant difference in student learning on array-based and linked lists as measured by the final exam. Students displayed half as much disengaged behavior during in-class labs and were five times more likely to ask for help from the teaching staff during in-class labs.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education – *computer science education.*

General Terms
Experimentation

Keywords
In-class labs; empirical computer science education; linear data structures; CS1.5

1. INTRODUCTION
Students struggle with the material in CSC216: Programming Concepts – Java, a second semester CS1.5 programming course for computer science majors and minors at North Carolina State University. While a large majority of students successfully complete CSC216 on a first or second attempt, many students report difficulty with the coursework throughout the semester.

ICER '15, August 9-13, 2015, Omaha, NE, USA.

ACM 978-1-4503-3630-7/15/08…$15.00.
DOI: http://dx.doi.org/10.1145/2787622.2787713

The prerequisite course of CSC216 is CSC116: Introduction to Programming – Java, a first semester introductory programming course taught in Java[1] with a use objects early, write objects late paradigm. CSC116 is an integrated-lecture lab with at most 33 students in each of seven or eight sections. The class meets twice a week for 110 minutes and one instructor and two TAs are available to help students. CSC216 moves students into two large lecture sections of 70-100 students. There is one instructor per section (sometimes the same instructor for both sections) and three to four TAs pooled for the two sections. A common request on end of semester evaluations for CSC216 is an increase in the amount of in-class programming practice similar to the level in CSC116.

Research has shown that active learning, defined by Freeman et al. [9] as "engaging students in the process of learning through activities and/or discussion in class, as opposed to passively listening to an expert," increases student learning through collaborative engagement with materials during class time [1, 5, 7, 9, 13, 15]. CSC216 currently incorporates a simplified version of the active learning technique, think-pair-share [7, 14], where the emphasis is on the pair and share. However, many students still struggle in the course, and we hypothesize that active learning practices that involve larger problems would increase student learning and engagement.

The study in this paper reports on the use of in-class laboratories, as an inverted or flipped classroom experience [15], for a unit on linear data structures in CSC216. *The goal of our research is to increase student learning and engagement through in-class laboratories on linear data structures.* We conducted a counter-balanced study on the use of in-class laboratories on two sections of CSC216 taught by the author at the same time on different days during the Fall 2014 semester. We found no significant difference in student learning on linear data structure topics; however, we found a large increase in student engagement measured by counts of off topic behavior and student interactions with teaching staff as reported by external observers. However, many of the interactions with students during in-class labs were focused more on the technology used in the course than on the lab topics.

We contribute to the growth of theory in computing education research by building on the foundations of theoretical work [17] in active learning [1, 5, 7, 9, 13, 15]. Additionally, our work builds on a foundation of Bandura's self-efficacy theory [2]. By reporting null-results for learning, we provide more data about the landscape of active learning interventions [21]. The contributions are:

[1] Oracle's Java may be found at: http://www.oracle.com/technetwork/java/index.html

- A replicable study methodology for assessing student learning and engagement when using in-class laboratory assignments.
- Additional data on the effectiveness of active learning activities, like think-pair-share exercises and in-class labs, on student learning and engagement.

This study is the first in a series of interventions focused on increasing student learning, engagement, and eventually completion through the incorporation of various active learning techniques and software engineering best practices into CSC216 coursework.

The rest of this paper is organized as follows: Section 2 describes related work; Section 3 describes the study methodology; Section 4 reports the results; Section 5 provides the threats to validity; Section 6 is a discussion of findings; and Section 7 concludes and presents future work.

2. RELATED WORK

A large meta-analysis on active learning in science, engineering, and mathematics found that active learning activities like think-pair-share [7, 14] and inverted or flipped classrooms support students learning, increase engagement, and reduce failure rates [9]. Freeman et al., [9] found that student performance on exams or concept inventories increased almost half of a standard deviation when using active learning as compared to traditional lecture. Kothiyal et al. [14] report an average 83% student engagement in lecture when using think-pair-share in CS1. These results are useful comparison points for our results.

Our delivery of in-class labs in CSC216 was modeled on the inverted or flipped classroom. Many CS researchers have investigated inverted classroom models in classes at various levels. Amresh et al., [1] conducted a preliminary study on the effectiveness of a flipped classroom in a CS1 for majors and non-majors. They found that students in the flipped sections of the course earned higher average scores. Student efficacy also increased, but the increase may be from other factors than just the course flip. Latulipe et al. [16] included lightweight teams and gamification in a flipped media computation class. Results show that lightweight teams enhanced student learning and increased course engagement. We used randomly assigned teams for in-class lab activities in CS1.5, but not the full lightweight team strategy.

Campbell et al., [5] reported on a study of inverting a CS1 course. They found that while fewer students attended the lectures in the inverted offering, more students completed the preparatory work including videos and quizzes, likely because the preparatory work counted for credit. Students also reported in a survey that they enjoyed the inverted model, but that they felt the course took more time. The authors found no significant difference in learning when compared with a traditional offering of the course. Horton et al., [12] continued the work by comparing a traditional and inverted CS1 course and reported similar pass rates, but a statically significant difference on final exam grade as a measure of student learning. We consider similar metrics for evaluation of our comparison of active learning lectures and in-class labs.

3. STUDY METHODOLOGY

The goal of our research is to increase student learning and engagement through in-class laboratories on linear data structures. We considered the following research questions:

RQ1: Do in-class laboratories on linear data structures increase student learning on linear data structure exam questions when compared to active learning lectures?

RQ2: Do in-class laboratories on linear data structures increase student engagement when compared with active learning lectures?

Several of the artifacts used for the study are available as a partial replication package [20] including the initial survey, observation protocol, and in-class laboratories and related materials [10]. Other materials, including informed consent, exam questions, and projects are available from the author by request.

3.1 Study Context

We conducted the study in CSC216: Programming Concepts – Java during the Fall 2014 semester. CSC216 is a second semester CS1.5 computer science course, which covers advanced object oriented programming, introductory software engineering, linear data structures, finite state machines, recursion, GUIs, sorting, and searching. The class meets twice a week for 75 minutes. Course grades are a combination of three tutorials [11], three two-part programming projects, in-class exercises, and three examinations. The author taught two sections of CSC216 during the Fall 2014 semester. Both sections met in large lecture halls with stationary desks and chairs.

In CSC216 students work with a number of tools to support the learning outcomes related to software engineering. Students develop assignments in the Eclipse Juno[2] integrated development environment using Java v1.7 with a suite of Eclipse plug-ins. Unit tests are written with JUnit v4[3] and coverage is measured by EclEmma[4], which uses the Jacoco[5] code coverage library. Static analysis tools, FindBugs[6], PMD[7], and CheckStyle[8], check for misuse of the Java language and styling problems. Student programming assignments are submitted for evaluation by pushing the project to our university's enterprise GitHub[9]. Student jobs are evaluated automatically with every push to GitHub by using the continuous integration server Jenkins[10]. Each student has a Jenkins job for their project and the job will build the student project, run the student's tests instrumented for coverage, run the static analysis tools, and run a suite of teaching staff unit tests, similar to Web-CAT [6]. Students are introduced to these technologies through a series of tutorials [11].

3.2 Study Participants

Students registered for their section of CSC216 on a first-come, first-served basis. Table 1 provides an overview of each section. We exclude counts on minority students due to low numbers that may lead to identification. The author solicited informed consent from students on the first day of class (NC State IRB #4169). Students opted into or out of the study and completed a survey. After the solicitation for participation, the author left the room and the author's Ph.D. student collected informed consents and

[2] Eclipse may be found at: http://www.eclipse.org/.
[3] JUnit may be found at: http://junit.org/.
[4] EclEmma may be found at: http://www.eclemma.org/.
[5] Jacoco may be found at: http://www.eclemma.org/jacoco/.
[6] FindBugs may be found at: http://findbugs.sourceforge.net/
[7] PMD may be found at: http://pmd.sourceforge.net/.
[8] CheckStyle may be found at: http://checkstyle.sourceforge.net/.
[9] GitHub may be found at: https://github.com/.
[10] Jenkins may be found at: https://jenkins-ci.org/.

surveys. To reduce bias, the author's Ph.D. student held all informed consents and surveys until final grade submission.

Table 1: Information about Participants

Metric	Section 001	Section 002
# Enrolled	85	102
Participants (completed course)	49	60
Dropped/Withdrawn (consenting only)	3	4
Women	9	10
Meeting Time	TH 2:05p-3:35p	MW 2:05p-3:35p

Students were given an introductory survey about their background and efficacy when completing informed consent [10]. These measures characterize the consenting populations and show that the participants are similar between the two sections. Table 2 shows the difference between Section 001 and Section 002 percentages on prior experience with using course technology as listed in Section 3.1. A positive number implies that Section 001 had a higher percentage of responses at that level of experience for the given tool and a negative response implies that Section 002 had a higher percentage. A difference of 0% shows that the populations were the same. For each possible Likert response, we characterized the prior experience measure by providing guidelines of the response's meaning in terms of the number of classes and work experience. For most responses, there was less than a 10% difference between sections. The only responses with a greater than 10% difference are shaded in gray. More students in Section 002 responded that they had "some" experience with Eclipse. Students in Section 001 had more students with "no" experience with the Eclipse Debugger and Static Analysis; however the Section 002 experience varies across the categories. These results overall show a relatively common set of prior experiences with course tooling.

Table 2: Prior Experience with Course Tooling

	None	Very Little	Some	Quite a Bit	Very Much
# of Classes	**None**	**< 1**	**< 2**	**< 4**	**> 4**
Work Exp.	**None**	**< 6 mos.**	**< 2 years**	**< 4 years**	**> 4 years**
Java	0%	0%	0%	0%	0%
Eclipse	9%	5%	-11%	-2%	-2%
Eclipse Debugger	15%	-8%	-5%	0%	-2%
Static Analysis	14%	-7%	-7%	0%	0%
Unit Testing	5%	-2%	-1%	-2%	0%
Code Coverage	8%	-4%	-3%	0%	0%
Version Control	-3%	7%	-3%	-3%	2%
Continuous Integration	-1%	-1%	0%	0%	2%

Another factor of student success is student self-efficacy or self-belief [2]. Bandera defines an efficacy expectation as "the conviction that one can successfully execute the behavior required to produce the outcomes" [2]. Computer science students know the steps to complete a programming assignment successfully, but may not believe that they have the ability to complete the task. Scott and Ghinea [19] found a relationship between a student's self-belief about their programming aptitude and their programming practice behavior, so self-efficacy is important for student success. We surveyed students on their confidence (Table 3). The self-efficacy questions are adapted from examples given by Bishop-Clark and Dietz-Uhler [4], but future studies may consider a preliminarily validated instrument by Scott and Ghinea [19]. Cells in gray are where there is more than a 10% difference between Sections 001 and 002 on their agreement with a given statement. Agreement with statements A, D, and E suggest confidence while agreement with statements B and C suggest a lack of confidence. The results show that both sections have a similar level of confidence. Section 002 had a higher percentage of students that strongly agree with the statement "I could learn programming and testing". A higher percentage of students in Section 001 were neutral on statements B and C, suggesting that they may have less self-efficacy about their programming skills.

Table 3: Programming and Testing Confidence

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
A	-17%	17%	0%	0%	0%
B	2%	-1%	13%	-13%	-1%
C	0%	-1%	10%	-11%	2%
D	4%	-21%	9%	9%	0%
E	2%	-14%	14%	3%	-5%

A: I am sure that I could learn programming and testing.
B: I am not good at programming and testing.
C: I am not the type to do well at programming and testing.
D: I have a lot of self-confidence when it comes to programming and testing.
E: Generally, I have felt secure about computer programming and testing.

3.3 Study Setup

The use of active learning activities, like think-pair-share exercises and in-class labs, are proven to increase student learning [9, 13]. To minimize the disadvantage to one group of students through the intervention of in-class labs, we used a counter-balanced study design so that each section of the course would have an opportunity to receive the intervention and we could measure student learning on each topic. Six lectures on linear data structures out of the 28 lectures in CSC216 were converted to in-class lab activities (a seventh in-class lab involving code inspection was also done, but it did not directly relate to linear data structures and is not considered in this paper). The linear data structures unit was selected because two of the topics, array-based and linked lists, are very similar and would work well in a counter-balanced study design as shown in Figure 1.

The gray boxes in Figure 1 represent class periods where students participated in the in-class lab intervention. The first and last labs, Lists (an overview of using the Java Collections Framework) and Iterators, were common for both sections. Section 001 received array-based list instruction as in-class labs and linked list instruction with active learning lectures. Section 002 received array-based list instruction with active learning lectures and linked list instruction as in-class labs. The details about active learning lectures are in Section 3.4 and details about the in-class labs are in Section 3.5.

Administration of Exam 1 occurred between the array-based list and linked list class periods. Two parts of the exam assessed student learning of array-based lists. Two parts of Exam 2 assessed learning of linked nodes and linked lists. The final exam included a question on both array-based lists and linked lists.

To assess student engagement, graduate students and a colleague participating in a graduate seminar on Teaching and Learning in Computer Science observed one or more of the class meetings. The observations were conducted on the eight class periods for array-based lists and linked lists – four classes for each section.

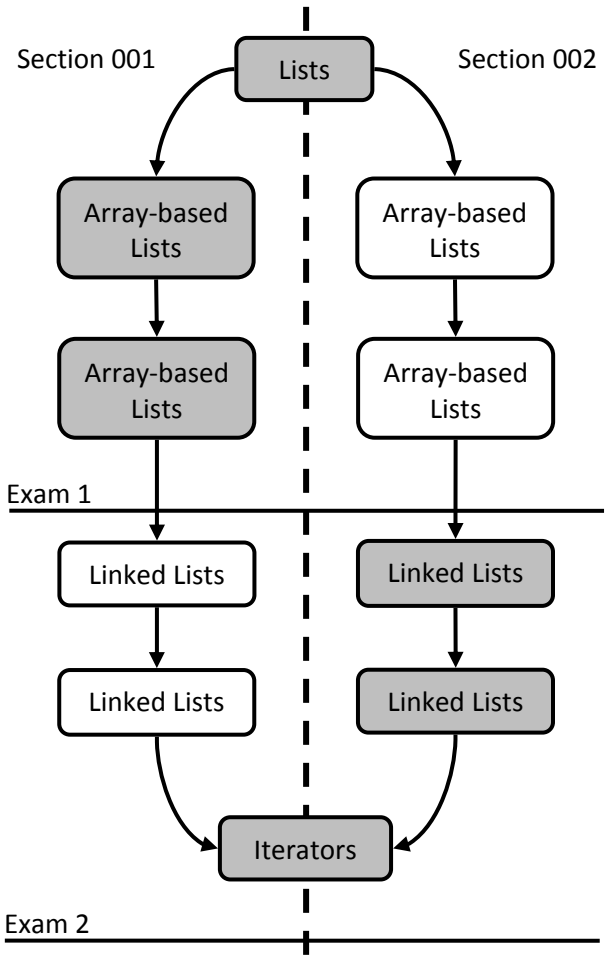

Figure 1: Study Design

Of the eight class periods, seven were observed. The provided observation protocol measured the off topic students and student's engagement with the teaching staff [10]. Off topic behavior was defined as "checking a non-course related website, working on a project or other assignment, checking their cell phone, etc." We asked observers to count the following items:

- number of instances of off topic behavior during lecture portions of the class
- number of instances of off topic behavior during exercise portions of the class
- number of students off topic during lecture
- number of students off topic during exercises
- number of times a member of the teaching staff is called for help
- number of students or student teams asking for help

3.4 Active Learning Lectures

CSC216 classes consist of lectures that are broken up with one or more simplified think-pair-share active learning exercises [7, 13]. Active learning exercises engage students with the materials just covered in lecture. Students are encouraged to work on the exercises with their neighbors (pair) and submit exercise answers through Google Forms (share). The Google spreadsheet backing the form updates automatically with new responses so the instructor can identify student misconceptions and address them after the exercise closes. The active learning exercises count as part of the exercise portion of the student's final grade. Students earn at least half credit for attempting the exercise.

The study control are the active learning lectures on array-based lists and linked lists. Each topic had two class periods devoted to instruction, with seven associated think-pair-share exercises per topic split across the two class periods. The first lecture provides a general overview of the data structure implemented as a list of integers. The second lecture provides instruction about how to implement the data structure using generic types.

3.5 In-class Laboratories

Students completed four of the six in-class laboratories depending on their section. Before the in-class lab, students were expected to watch a short 10-15 minute video about the topic of the lab activity. As students entered the classroom, they joined their groups by finding the paper tent on the desk with their team number on it. The class meeting began with a short lecture introduction to the lab activity, and then students started the lab activity. The class concluded with a reflection exercise and a reminder to complete the activity outside of class. The in-class labs were intended to be counted in the exercise portion of the student's grade, but were excluded by TAs during exercise grade calculation.

Students were randomly placed on teams of three and were assigned a GitHub repository for their in-class lab work separate from their GitHub repositories for submitting projects. Students with missing teammates were reassigned to another team during class to minimize students working on the in-class labs alone.

The design of the in-class laboratories involved development exercises using the software engineering tools for the class. Each lab, except the first lab, was broken into small tasks. The first lab was a Power Point slide, but now the lab is available as a Google doc like all the other labs. At the end of each task, the instructions stated students should run their tests, comment their code, and push to GitHub. All lab materials, including slides, videos, and lab activities are available at [10].

3.5.1 Java Collections Framework In-class Lab

The goal of the Java Collections Framework lab is the creation of a suite of unit tests appropriate for testing a linear data structure. By writing unit tests in the first in-class lab, the students practice test-driven development and can use the tests to evaluate later implementations of linear data structures. Students started with an Eclipse project containing a jar'd implementation of an `ArrayList` of Strings. A unit test skeleton and Javadoc of the `ArrayList` implementation contained the instructions for writing sufficient unit tests for 100% condition coverage of the `ArrayList` implementation. The instructions for this in-class lab were initially provided on a slide projected for both sections. Students struggled with projected instructions, so future labs had more detailed instructions. The Java Collections Framework in-class lab has been revised for future offerings.

Table 4: Comparison of Exam Items

Exam Item	Total Points	Section 001			Section 002			W	p-value	Lower CI	Upper CI	Diff. in Loc.
		Mean	SD	Med.	Mean	SD	Med.					
E1 P4 #8	5	3.63	1.56	5	4.35	1.45	5	1064.5	< 0.010	-0.99	-3.7E-6	-5.3E-5
E1 P4 #9	5	4.18	1.07	5	4.57	1.09	5	1167.0	0.016	-5.7E-5	-3.8E-5	-5.0E-5
E1 P4 #10	5	2.63	2.40	2	3.45	2.18	5	1257.5	0.149	-9.9E-6	8.2E-5	-4.7E-6
E1 P4	15	10.45	3.97	10	12.37	3.74	14.5	1054.5	< 0.010	-3.99	-3.3E-5	-1.99
E1 P5	15	17.76	4.00	20	18.25	4.09	20	1294.0	0.233	-4.9E-5	3.5E-5	-5.1E-5
E2 P3	16	8.43	5.85	8	11.80	6.41	16	955.0	< 0.010	-7.99	-4.5E-5	-3.99
E2 P5	20	11.80	4.14	12	12.58	4.21	13	1257.5	0.412	-2.99	0.99	-0.99
E3 Array	10	8.31	2.45	9.5	8.46	2.49	9.5	1096.5	0.313	-0.50	5.9E-5	-1.8E-5
E3 Linked	10	8.36	2.53	9.5	8.81	2.45	10	1069.5	0.221	-0.49	1.8E-5	-4.4E-5
Exam 3	105	85.02	29.17	94.5	87.23	28.92	97.5	1323.0	0.372	-6.00	2.00	-1.50

3.5.2 In-class Lab 1 for Lists

The goal of Part 1 of the `ArrayList` and `LinkedList` labs is to write an array-based or linked list of Strings. Students start by copying and refactoring their tests from the Java Collections Framework lab into a new test file for verifying their `List` implementation. Students then work on standard list functionality, state, constructor, `size()`, `get()`, `add()`, `remove()`, and `set()`, with reminders to document their code, run their tests, push to GitHub, and switch drivers.

After implementing the major functions of the `List` class, students evaluate their test coverage on their solution and work on a stretch goal of writing a program that uses the `List`.

3.5.3 In-class Lab 2 for Lists

The goal of Part 2 of the `ArrayList` and `LinkedList` labs is to write a generic list. Students start from the previous lab's code and refactor both their tests and their implementation to work for any type of object.

3.5.4 Iterators In-class Lab

The goal of the Iterators lab is for students to implement a `LinkedList` by extending `AbstractSequentialList`. The standard `List` methods in `AbstractSequentialList` are implemented in terms of an `Iterator`. Students must write a custom Iterator for `AbstractSequentialList` to work. Students continue to use their tests from the Java Collections Framework lab to verify their solution.

4. STUDY RESULTS

The following sections outline results of the in-class lab activities on student learning of linear data structures and student engagement.

4.1 Learning on Linear Data Structures

Three written examinations measured student knowledge on linear data structures. We only report the summary exam data for consenting students.

We administered Exam 1 between the array-based lists and linked lists portions of the linear data structures unit. Two parts (Part 4 and 5) of Exam 1 assessed student knowledge on array-based lists. Part 4 assessed the ability of students to trace how the contents of an `ArrayList` are modified when passed to a mystery method. There were three questions in Part 4; each question was evaluated separately and all three were evaluated together. Part 5 assessed the ability of students to write a method for an `ArrayList` class.

Exam 2 had two parts (Part 3 and Part 5) assessing student knowledge on linked lists. Part 3 evaluated students' ability to work with linked nodes. Students wrote code to transform a before picture of linked nodes into an after picture of linked nodes. Part 5 assessed the ability of students to write a method for a `LinkedList` class.

Exam 1 and Exam 2 have similar structure, but the questions differed between sections to minimize sharing of knowledge. For example, Part 5 for Section 001 asked students to write `indexOf()`; Part 5 for Section 002 asked students to write `lastIndexOf()`. Both sections took a common final exam, called Exam 3. The final exam included a question on implementing an array-based stack and a linked list-based queue. The final exam grade was also considered as a whole to determine any difference in learning between sections.

Each exam part and sub question were tested for normality using the Shapiro-Wilk Normality Test in the R Project for Statistical Computing [18]. All exam items were nonparametric with p-values < 0.01. Due to the nonparametric distribution of the data sets, a two-sample, non-paired, two-sided Wilcoxon test was used to compare the scores on each exam item between the two sections using R. A two-sided test will determine if there is a difference between the distributions in each section. Table 4 summarizes the results.

The gray cells in Table 4 show the four exam items where the distributions of the sections are different at a statistically significant confidence level of 95%. The first three gray cells are for Part 4 of Exam 1, which was a question about using the data structure as a client and not about implementing part of the data structure. The last gray cell is for Exam 2 Part 3, which was a question about manipulating linked nodes.

For the statistically significant results in Table 4, we ran a one-sided Wilcoxon test to determine if the intervention leads to an increase in exam item scores for the section that received the intervention. Table 5 summarizes the results.

Table 5: One-sided Wilcoxon Test on Significant Exam Items

Exam Item	Test	p-value	Lower CI	Upper CI
E1 P4 #8	001 > 002	0.999	-1.0	Infinity
E1 P4 #9	001 > 002	0.992	-3.0E-5	Infinity
E1 P4	001 > 002	0.996	-3.00	Infinity
E2 P3	001 < 002	< 0.01	-Infinity	-4.01E-5

Our results show that Section 001, which completed the in-class labs on array-based lists did not outperform Section 002 on Exam 1 Part 4. When testing the hypothesis that Section 001 scores were greater than Section 002 scores, the p-value was strongly not significant. A one-sided Wilcoxon test that switches the

alternative hypothesis does lead to statistical significance, which means that Section 002 outperformed Section 001 on Exam 1 Part 4. The only place where the alternative hypothesis of a section with the in-class lab intervention outperforming the other section on the related exam material was for Exam 2 Part 3 where Section 002 outperformed Section 001. However, alone, that result is not strong enough to demonstrate higher gains in learning on topics taught using in-class lab activities. Our results show that there was no major difference in student learning as measured by exams when comparing active learning lectures with in-class laboratories.

4.2 Engagement

We measured engagement through classroom observations by graduate students and a colleague participating in a seminar on Teaching and Learning. Each observer used an observation protocol [10]. All students attending class during the observation were observed. The observation protocol included no identifying information about students and consenting students were unknown during the observation period.

Table 6 summarizes the observation counts for the number or average number of instances of off topic behavior during lecture and exercise portions of the class and the number of instances when students engaged with the teaching staff. Each observer used the observation protocol in a slightly different manner. Therefore, each observation period will be described in the following sections followed by a discussion of common metrics and themes. The three observations that considered the average over five-minute intervals, Observations 2, 4, and 8, were completed by the same observer and the summation of the off-topic observations would significantly skew results. The average is more representative of the values reported by other observers.

Table 6: Observation Summary

Obs.	Class Type	# Off Topic Lecture	# Off Topic Exercise	Questions of Teaching Staff
1	Lab	5	7	32
2	Lec.	62	49	12
3	Lab	10	43	50
4	Lec.	46	16	----
5	Lec.	----	----	----
6	Lab	5	10	33
7	Lec.	52	54	2
8	Lab	16	5	-----
Lab Average		9	16.3	38.3
Lec. Average		53.3	39.7	7
Lec. / Lab		5.9	2.4	0.2

4.2.1 Observation 1: Array-Based List 1 In-Class Lab

Section 001 completed the array-based list 1 in-class lab. The observer recorded counts every 5 minutes in two groups: 1) general disengagement and 2) those looking at a Google document. The students in the second group were likely on task since that day's activity was provided in a Google document, so they are not considered in the summary.

During the initial lecture in the first five minutes of class, there were five instances of disengaged behavior. During the in-class lab, there were seven instances of disengaged behavior. Students asked the teaching staff for help 32 times during the class period.

4.2.2 Observation 2: Array-Based List 1 Active Learning Lecture

Section 002 received a standard active learning lecture on the first half of the materials on array-based lists. The observer focused on the number of instances of on and off topic behavior in five-to-ten-minute windows. The window was then marked as lecture or exercise. In a class with attendance of 78 students and with 62 visible laptop screens, an average of 61 students were off topic during lecture and an average of 50 students were off topic during the active learning exercises. The teaching staff was asked for help 12 times during the class period.

4.2.3 Observation 3: Array-Based List 2 In-Class Lab

Section 001 completed the array-based list 2 in-class lab. Two observers attended the class and each recorded notes on one-half of the class. During the initial five-minute lecture portion of the class, there were 10 instances of disengaged behavior. There were 43 instances of disengaged behavior during the in-class lab. Students or student teams asked for help over 50 times.

4.2.4 Observation 4: Array-Based List 2 Active Learning Lecture

Section 002 received a standard active learning lecture on the second half of the materials on array-based lists. The observer focused on an estimate of the number of off topic students by observing screens visible from his spot and creating a class wide estimate of the number of students off topic during five-minute windows. In a class with attendance of 72 students, 65 total screens, and 30 screens used for the estimate, an average of 46 students were disengaged during the lecture portion of the class. An average of 16 students were disengaged during the think-pair-share exercises. Question counts were not recorded.

4.2.5 Observation 5: Linked List 1 Active Learning Lecture

Section 001 received a standard active learning lecture on the first half of materials on linked lists. No observer attended.

4.2.6 Observation 6: Linked List 1 In-Class Lab

Section 002 completed the linked list 1 in-class lab. The observer recorded 5 instances of disengaged behavior during the short introductory lecture and 10 instances of disengaged behavior during the activity and reflection portion of the class. The observer grouped together the likely 25 or more students who were observed as disengaged during the reflection portion of the class as a single incident. There were 33 instances where the teaching staff was called for help.

4.2.7 Observation 7: Linked List 2 Active Learning Lecture

Section 001 received a standard active learning lecture on the second half of materials on linked lists. The observer noted that he could only see half of the class during this observation. Reported numbers are doubled with the assumption that the observed half is indicative of the whole. There were and estimated 52 instances of disengaged behavior during the lecture portion of the course by an estimated 20 unique students. During the exercises, there were an estimated 54 instances of disengaged behavior by an estimated 18 unique students. Only two students asked for help during the class period.

4.2.8 Observation 8: Linked List 2 In-Class Lab

Section 002 completed the linked list 2 in-class lab. The observer counted 57 students with 50 laptops, 26 of which were visible

during the observation. The number of disengaged students was estimated from the visible laptops. An average of 16 students were off topic during the lecture portion of the course and an average of 5 students were disengaged during the in-class lab portion of the course. Question counts were not recorded.

4.2.9 Observation Summary

For active learning lectures, students were over five times more likely to display off topic behavior during lecture portions of the class and over two times more likely to display off topic behavior during the exercise portion. During in-class labs, students were over five times more likely to engage with the teaching staff by asking questions.

These results confirm the instructor's reflection on each of the class periods: students engaged with their peers, engaged with the lab activity, and asked more questions of the teaching staff than then in the active learning lectures. However, the observers just counted the number of questions and interactions. The missing piece of the observation is the content of the questions.

Many of the questions, especially for the early in-class labs like the un-observed Java Collections Framework lab, involved technology. The first Java Collections Framework in-class lab had a problem with the provided library. The provided library was compiled and jar'ed using Java 1.8, and many students only had Java 1.7 installed. Due to the volume of questions, the instructor was unable to resolve the issue until the very end of class. The other section had fewer library issues, but had many tool and technology questions. Another common question during the first few in-class lab activities was how to handle a fast-forward error from GitHub. These questions led to the creation of a "Troubleshooting" section at the end of each lab so that the instructor could quickly refer students to the section and move on to answer other questions.

The benefit of the in-class lab experience was that students were able to resolve or consider alternative solutions to problems quickly, especially problems associated with tooling, through the help of their peers and the teaching staff. However, since many of the student questions focused on tooling, students may not have engaged deeply with the covered course topics due to the difficulties with the tools. As the labs progressed, students asked more questions about the lab topics, but no counts of specific question types were recorded during the observed classes. While students were more engaged, as measured by off-topic behavior and teaching staff interactions during in-class labs, the engagement was not solely on the course topics.

5. THREATS TO VALIDITY

Due to the nature of classroom research, there are several threats to validity that constrain the generalizability and application of our results [4, 8].

5.1 External Validity

External validity describes the generalizability of our results for the study population [8]. Our study was restricted to two sections of the same course taught by the same instructor in the same semester and even at the same time of day. While these study constraints allowed for an additional level of control and an increase of internal validity, the constraints do limit the generalizability of this work to other CS1.5 classes, courses, and institutions. Due to the nature of cohorts, there may be differences between results in fall and spring offerings of the same course. Replications of the study in future CSC216 classes, other CS1.5 courses, and at other institutions would increase the generalizability of the work. A replication package [20] with the study materials is available upon request, but some materials, including labs, initial survey, and observation protocol, that would not affect future studies are available [10].

5.2 Internal Validity

Internal validity is maximized through the reduction of bias and concerns the quality of our conclusions [8]. The study does have selection bias: students selected their own sections. However, an initial survey of students, conducted as part of the collection of informed consent, shows that the characteristics of the sections were roughly the same. Table 2 and Table 3 show each section had participants with similar characteristics, which minimizes sample bias.

There are several possible confounding factors. Students given one intervention may have shared that material with students taught using the control. Additionally, since only a portion of the course incorporated in-class labs, any effect may have been too small to measure. Additional studies with more in-class labs that span the breadth of course topics appropriate for lab exercises would increase internal validity of future studies.

We minimized differential attrition bias by including consenting participants that soft dropped the course. Students that dropped or withdrew did not complete any coursework evaluated in the study.

There is the possibility of experimenter bias because the author created the study and taught both sections of the course. The author tried to reduce experimenter bias by requesting that her Ph.D. student hold all informed consents until the end of the semester so that she would not know who consented to participate in the study. Due to the nature of the classes in the treatment and control, there were differences in how the author presented the materials to each section, but she tried to remove her preferences for lecture success from discussions with the students. Future studies will include other faculty to reduce experimenter bias.

5.3 Construct Validity

Construct validity describes how well the underlying concept of interest, in our case student learning and engagement, are empirically investigated [8]. Exam questions measured student learning. For the first two exams, each section had similar, but different questions. On exam 1, Section 001 wrote the `indexOf()` method and Section 002 wrote the `lastIndeOf()` method for an array-based list. The questions were similar, but iterating through a list in reverse may be conceptually harder. That means the exam questions on Exams 1 and 2 may not equally assess student learning of the topic. Students took a common final, so those questions provide a common comparison point.

Another concern is the exam questions and the grading rubrics themselves. The exam questions and the rubric may not fully evaluate student learning. The department's assessment coordinator has assessed the final exam, and there is no current concern that the exam is not measuring student learning.

The initial survey asked students about their prior experience with tooling and about their self-efficacy when programming and testing [4]. There were additional questions about enjoyment [4], ways of learning, goals, and demographics. The addition of course and time information on the tooling question may provide additional confusion to students about if the relationship is an "and" or an "or" relationship. However, the results of that question were what we expected from courses typically made up of traditional aged college freshman and sophomores. The

questions on efficacy and enjoyment were pulled from literature [4], but are not to our knowledge validated instruments. Future studies may consider a preliminarily validated instrument by Scott and Ghinea [19]. All other questions were generated by the author and may be flawed instruments. Further validation of the survey instrument is needed to determine if the questions measure what they are supposed to measure, but those questions do not influence the results of this study.

The intention of the observation protocol was to measure how many times and how many students were off task. Additionally, the protocol measured the number of times the teaching staff engaged with students about course material. However, inconsistent use by observers is a construct threat. Many, but not all, observers measured the counts at five-minute intervals. Future iterations of the protocol will create a timeline for counts and will include summary information like the number of students in attendance. That instrument will allow for better summary results from multiple observations and multiple observers.

6. DISCUSSION

The goal of our research is to increase student learning and engagement through in-class laboratories on linear data structures.

Research question 1 asked do in-class laboratories on linear data structures increase student learning on linear data structure exam questions when compared to active learning lectures? Our results show that in-class laboratories on linear data structures did not lead to an increase in student learning over active learning lectures. Student learning when using in-class labs and active learning lectures is the same as measured by the linear data structure questions on the final exam. Ultimately, the result is not unexpected since both think-pair-share exercises as used in the active learning lectures and in-class labs are both active learning techniques that both lead to increased learning. Overall, the results suggest that using in-class laboratories does no harm to student learning. Future work may formalize the team portion of the in-class labs to use lightweight teams [16], which may increase student learning. Additional studies will strengthen our knowledge.

Research question 2 asked do in-class laboratories on linear data structures increase student engagement when compared with active learning lectures? The observations show that most students engaged with the material during in-class lab sections while a large portion of students were off topic during active learning lectures. Additionally, over five times as many students or student teams asked questions during in-class labs than during active learning lectures. Due to inconsistent and estimated measures during the observations, we cannot attempt any statistical analyses on the data. The raw numbers are highly suggestive that in-class labs were more engaging. However, many of the student questions involved tooling used for the in-class labs rather than questions on in-class lab topics. While students were more engaged during in-class labs, the engagement was not solely on the lab topics, which may contribute to the null-results on student learning. Future studies with better observation protocols, including categorization of student interactions by teaching staff, can further answer our research question.

Prior research on active learning has shown an increase in completion rates [9], where the completion rate is the percentage of students who pass the course. Grade distributions for courses at NC State are protected data, so we cannot compare or comment on how completion rates for the Fall 2014 offerings of CSC216 compare to prior offerings or other core undergraduate computer science courses at NC State. While we cannot compare publically against our own historical data, we can compare with reported completion rates and use these measures as a baseline in future studies that build on this work. Bennedsen and Caspersen [3] report a 67% pass rate for CS1 courses in 2007 and Watson and Li [22] report a similar completion rate of 67.7% for CS1 courses in 2014. We expect that a CS1.5 course would have a similar completion rate due to an overlap with traditional CS1 topics. Freeman et al., [9] found a completion rate of 78.2% with active learning compared to 66.2% completion rate for traditional lecture courses. Seventy-two percent of students completed CSC216 in Fall 2014 with a C or higher. CSC216's completion rate of 72% is lower than the completion rate for active learning classes reported by Freeman et al. [9], but not as low as the completion rate for traditional lecture courses. The completion rate for CSC216 is higher than the completion rates from CS1 literature [3, 22]. Additional reportable data about prior semesters and comparison with future semesters will identify if we are making progress toward increasing the rate of student completion in CSC216.

7. CONCLUSIONS AND FUTURE WORK

Overall, the use of in-class labs was successful in maintaining student learning and increasing student engagement. There were several lessons learned from incorporating in-class labs into CSC216. Students were expected to watch videos about the lecture material before attending class to complete the in-class lab. Based on questions received and the number of students watching the videos at the start of the class, most students did not prepare adequately for class. One solution is to have students take a quiz on the material for a grade [5]. Another solution is to restructure the array-based list and linked list lectures so that the first class period on the topic will be a lecture and the second class period will be devoted to the in-class lab activity for implementing a generic version of the data structure.

We may not have seen gains in student learning since only six lectures of a 28-lecture course were changed to in-class labs. Additional in-class labs will be developed for advanced OO, stacks and queues, FSMs, recursion, and GUIs. Future work will also consider the identification and transfer of the in-class lab software engineering best practices to out-of-class assignments and future coursework.

The introduction of in-class labs addressed a student request in evaluations for more time programming during class, but we would still like to increase student learning with a long term goal for increasing the completion rate. A departmental task force of CSC216 instructors and other undergraduate leadership is working on increasing student support and moving the course to a lab-based delivery mechanism. In-class labs developed or refined for the next academic year will be assessed as a baseline for comparison to the move to a lab-based course in AY16-17.

8. ACKNOWLEDGMENTS

Thanks to Brittany Johnson for survey collection and storage, Jordan Connor for help with data entry and review, and students and colleagues in CSC801-006 for their observations. Thanks to the students who participated in the study. Funding for the study is provided by a NCSU Office of Faculty Development grant and by a Google CS Engagement Award (TFR15-00445).

9. REFERENCES

[1] A. Amresh, A. R. Carberry, J. Femianai, "Evaluating the Effectiveness of Flipped Classrooms for Teaching CS1,"

Frontiers in Education Conference, Oklahoma City, OK, USA, 23-26 Oct. 2013, p. 733-735.

[2] A. Bandura, "Self-efficacy: Toward a Unifying Theory of Behavioral Change," *Psychological Review,* vol. 82, no. 4, pp. 191-215, 1977.

[3] J. Bennedsen and M. E. Caspersen, "Failure Rates in Introductory Programming," *SIGCSE Bulletin*, vol. 39, no. 2, pp. 32-36, 2007.

[4] C. Bishop-Clark and B. Dietz-Uhler, *Engaging in the Scholarship of Teaching and Learning*, Stylus Publishing, Sterling, VA, 2012.

[5] J. Campbell, D. Horton, M. Craig, P. Gries, "Evaluating an Inverted CS1," Proceedings of the 45th ACM Technical Symposium on Computer Science Education, Atlanta, GA, USA, March 3-8, 2014, p. 307-312.

[6] S. H. Edwards, "Improving Student Performance by Evaluating How Well Students Test their Own Programs," *Journal on Educational Resources in Computing*, vol. 3, no. 1, September 2003.

[7] R. M. Felder and R. Brent, "Active Learning: An Introduction," *ASQ Higher Education Brief*, vol. 2, no. 4, August 2009.

[8] S. Fincher and M. Petre, eds., *Computer Science Education Research*, Taylor & Francis, The Netherlands, Lisse, 2004.

[9] S. Freeman et al., "Active Learning Increases Student Performance in Science, Engineering, and Mathematics," *Proceedings of the National Academy of Sciences in the United States of America*, vol. 111, no. 23, p. 8410-8415, June 10, 2014.

[10] S. Heckman, "CSC216 In-class Lab Study Replication Package," [Online]. Available: http://people.engr.ncsu.edu/sesmith5/216-labs/csc216_labs.html

[11] S. Heckman, J. Perry, J. King, E. Gehringer, A. Meneely (2014, August 11) *CS1.5 Tutorials* [Online]. Available: http://courses.ncsu.edu/CS1.5/common/tutorials/.

[12] D. Horton, M. Craig, J. Campbell, P. Gries, D. Zingaro, "Comparing Outcomes in Inverted and Traditional CS1," Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education, Uppsala, Sweden, June 23-25, 2014, pp. 261-266.

[13] A. King, "From Sage on the Stage to Guide on the Side," *College Teaching*, vol. 41, no. 1, Winter 1993, p. 30-35.

[14] A. Kothiyal, R. Majumdar, S. Murthy, S. Iyer, "Effect of Think-Pair-Share in a Large CS1 Class: 83% Sustained Engagement," Proceedings of the 9th Annual Conference on International Computing Education Research, San Diego, CA, USA, August 12-14, 2013, pp. 137-144.

[15] M. J. Lage, G. J. Platt, M. Treglia, "Inverting the Classroom: A Gateway to Creating an Inclusive Learning Environment," *The Journal of Economic Education*, vol. 31, no. 1, p. 30-43, January 2000.

[16] C. Latulipe, N. B. Long, C. E. Seminario, "Structuring Flipped Classes with Lightweight Teams and Gamification," Proceedings of the 46th ACM Technical Symposium on Computer Science Education, Kansas City, MO, USA, March 4-7, 2015, pp. 392-397.

[17] L. Malmi et al., "Theoretical Underpinning of Computing Education Research – What is the Evidence?," Proceedings of the 10th Annual Conference on International Computing Education Research, Glasgow, United Kingdom, August 11-14, 2014, pp. 27-34.

[18] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, http://www.R-project.org/, 2013.

[19] M. J. Scott, G. Ghinea, "Measuring Enrichment: The Assembly and Validation of an Instrument to Assess Student Self-Beliefs in CS1," Proceedings of the 10th Annual Conference on International Computing Education Research, Glasgow, United Kingdom, August 11-14, 2014, pp. 123-130.

[20] F. Shull, V. Basili, J. Carver, J. C. Maldonado, G. H. Travassos, M. Mendonça, and S. Fabbri, "Replicating Software Engineering Experiments: Addressing the Tacit Knowledge Problem," Proceedings of the 2002 International Symposium on Empirical Software Engineering, Nara, Japan, October 3-4, 2002, pp. 7-16.

[21] A. Vihavainen, J. Airaksinen, C. Watson, "A Systematic Review of Approaches for Teaching Introductory Programming and Their Influence on Success," Proceedings of the 10th Annual Conference on International Computing Education Research, Glasgow, United Kingdom, August 11-14, 2014, pp. 19-26.

[22] C. Watson, F. W. B. Li, "Failure Rates in Introductory Programming Revisited," Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education, Uppsala, Sweden, June 23-25, 2014, pp. 39-44.