Role of Larger Software Artifacts in Introductory Computer Science Courses

Oleg Krogius, Thomas B. Horton, Mark S. Sherriff
krogius@umich.edu, horton@cs.virginia.edu, sherriff@cs.virginia.edu

Abstract - **This paper compares the effectiveness of two approaches that can be used to teach concepts in introductory courses such as CS1 and CS2 – a conventional lecture-based approach and one using larger software programs (artifacts) with accompanying guided exercises. Our assessment includes measures of students' self-confidence as well as a measurement of their knowledge of the topics used in this study: inheritance and iterators. Finally, we consider some generalizations that can be made about these treatments and how well they perform.**

Index Terms – Introductory Courses; Software Artifacts

INTRODUCTION

Active learning techniques have become increasingly popular in computer science education, and with good reason. Evidence shows that students often have better learning experiences in environments that use active learning techniques [1].

In computer science education, one popular form of active learning technique is learning by case study. Specifically, in this case, a large pre-made software system. Larger software programs are often more interesting as they are easier to relate to the world outside of academia, offer more learning opportunities beyond the basic content illustrated in a small example given in a lecture, and give students experience with dealing with larger projects. However, these programs may be intimidating to introductory students or lead to undesired confusion.

Artifacts or case studies have been used in computer science education in the past. A curriculum revision at our university that emphasized software engineering in early courses used an artifact for a code-reading exercise in the first course [2]. The Computer Science AP exam has made use of case studies since 1994-95. Large artifacts have also been a used in software engineering courses at the undergraduate and graduate level, such as the work described by Tomayko [3].

In this study, we examine the use of a large software system as a teaching tool in a CS2 course and compare its effectiveness against that the more conventional lecture instructional approach.

BACKGROUND

This experiment was conducted in the fall of 2008 in our CS2 course at University of Virginia. The course offers an introduction to software engineering using Java. Around 170 undergraduate students take this course during the fall. The course is required for all Computer Science, Computer Engineering, Electrical Engineering, and Systems Engineering majors. The course is roughly one-third Systems Engineering students, one-third Computer Science, Computer Engineering, and Electrical Engineering, and one-third of other majors from the School of Engineering and Applied Science.

The particular larger software artifact used for this experiment was RealEstate, a game similar to Monopoly developed by researchers at North Carolina State University to be used as a teaching example [4]. This particular program has been successfully used in the past couple of semesters both as a laboratory activity and as a separate homework assignment.

EXPERIMENTAL PROCEDURE

Given that the control treatment involved conventional lectures with examples, it was not practical to break up the class into two groups, thus giving RealEstate to one withholding lectures, and vice-versa with the other. Instead, the experiment was repeated twice with two different (albeit similar) topics, varying the order of the treatments (lecture and large software system). First, the experiment examined how lectures and the large software system impacted student learning of inheritance, and the second, how well students learned iterators. The order of treatments was reversed for iterators to examine how much of a difference each treatment made without significant bias to order.

I. Outline

Students were given surveys to measure their knowledge (both self-perceived confidence as well as questions to demonstrate actual knowledge) about the topics around each treatment; thus, it is possible to compare how each treatment impacted each student. The self-confidence questions ("How comfortable do you feel about X?") allowed the student to respond on a scale of one (not comfortable) to five (very comfortable). The remaining knowledge questions were graded and converted to a single floating point score from zero to one. Generally each question (or logical subpart) was

978-1-4244-6262-9/10/$26.00 ©2010 IEEE

graded on an integer scale of zero to two, and the individual questions (or logical subparts) were averaged to produce this single score.

Thus, the overall timeline of the experiment went as follows:

1. Initial survey on inheritance
2. Lecture on inheritance
3. Follow-up survey on inheritance and initial survey on iterators
4. RealEstate lab activity (involving both inheritance and iterators)
5. Final survey on inheritance and follow-up survey on iterators
6. Lecture on iterators
7. Final survey on iterators
8. Additional survey allowing free-form comments and to measure other factors

With this approach, we were able to gauge how much of an improvement is made by each lecture or the RealEstate activity. The last survey also allowed for students to give free-form comments about the experiment (which was fully explained only then) and to answer a few other questions to see how well-versed they were in the topics before the course, how interested the topics were, and how they found RealEstate program as a learning tool.

II. Surveys

The surveys were given in lab electronically. Students were informed that their course grades would not be impacted by the surveys in order to facilitate honest answers. Moreover, the surveys after lectures specifically asked if the student attended the lecture (if not, the results were discounted) without any penalty to the student.

The contents of the surveys on each topic did not change significantly from different administrations (same questions, but some words were replaced – e.g. instead of asking about Books the subject was changed to Classes). This was done to minimize the bias from using different questions. The two surveys can be found in Figures 1 and 2.

CS 201 – Inheritance Survey Questions

Please answer the following questions to the best of your ability. The answers to these questions will not affect your grade in this class.

1. How comfortable are you with inheritance? (1 – not at all, 5- very)
2. Consider the following code:

```
class Book {
    public String getName() { … }
}
class Novel extends Book {
   public String getGenre() { … }
}
class Thriller extends Novel { … }

Book a = new Book();
Book b = new Novel();
Novel c = new Novel();
Book d = new Thriller();
Novel e = new Thriller();
Thriller f = new Thriller();
```

Now, which of the following statements work (i.e. compile and run – don't worry about whether they do something useful)? If you don't know, leave the question blank.

a) a.getGenre();
b) b.getGenre();
c) c.getGenre();
d) d.getGenre();
e) e.getGenre();
f) f.getGenre();
g) Novel n1 = (Novel) a;
h) Novel n1 = (Novel) b;
i) Novel n1 = (Novel) d;
j) Novel n1 = (Novel) f;
k) Thriller t1 = (Thriller) c;
l) Thriller t1 = (Thriller) d;

FIGURE 1
INHERITANCE SURVEY.

RESULTS AND DISCUSSION

Almost all the students participated in the study – there are responses from 151 different individuals. However, not all of these stayed with the course, and a few students only responded to the first few surveys. Nevertheless, there are generally over a 100 responses on both sides of any treatment giving adequate information for comparison of how well a treatment worked.

I. Effects on Self-Confidence

The survey averages on self-confidence provide some information as to how the students believe they fared on the surveys without knowing any of the results. The survey averages on self-confidence can be found in Table 1.

From this data is it evident that both methods help students become more confident about their knowledge. However, it is interesting to note that lectures had a significantly greater impact (approximately double) on self-confidence than RealEstate. This may be explained by students being more familiar with this type of instruction from other classes and greater accessibility of lectures (as shown by the last survey results).

A Student's t-test (two-tail) produced probabilities of $5x10^{-16}$, $5x10^{-5}$, $4x10^{-21}$ for comparing *Initial* to *After Lecture*, *After Lecture* to *After RealEstate*, and *Initial* to *After RealEstate* respectively for inheritance. Similarly for iterators, the probabilities produced were < .005 for

comparing *Initial* to *After RealEstate*, *After RealEstate* to *After Lecture*, and *Initial* to *After Lecture* respectively.

CS 201 – Iterators Survey Questions

Please answer the following questions to the best of your ability. The answers to these questions will not affect your grade in this class.

1. How comfortable are you with iterators? (1 – not at all, 5 – very)
2. What is an iterator? A one or two sentence answer is all we're looking for. If you don't know, leave the question blank.
3. What is the main benefit of iterators? (i.e. Why use them rather than many other alternatives?) A one or two sentence answer is all we're looking for. If you don't know, leave the question blank.
4. Is the following code likely to work? If so, what does it do? If not, what is the problem?

```
Iterator it = arraylist1.iterator();
while ( it.hasNext() ) {
    if ( it.next() instanceof Student ) {
        Student s = (Student)it.next();
        System.out.println(s.getName());
    }
}
```

5. True/False: An iterator can only iterate over similar (i.e. same class) objects.
6. True/False: You can create your own iterators for a class you write.

FIGURE 2
ITERATORS SURVEY

II. Effects on Measured Knowledge

The actual knowledge assessment provides a deeper understanding of the phenomenon show in section 4.1. As shown from the scores in Table 2, it seems that RealEstate did not make significant improvement to knowledge despite improving self-confidence. This result (and student comments) indicates that a follow-up learning exercise based on an artifact only helps students reinforce their concepts rather than establish new understanding. While some of this reinforcement may be positive and help students with later retention (something not addressed in this study), it seems like this is a relatively small benefit. Conversely, showing a larger software artifact in the initial stages of a learning activity may produce a much more significant gain on understanding compared to lecture.

TABLE I
SELF ASSESSMENT (SCALE: 1-5)

	Mean	**Standard Dev.**
Inheritance (Conventional then Real Estate)		
Initial	2.22	1.17
After Lecture	3.11	1.07
After Real Estate	3.48	1.05
Iterators (RealEstate then Conventional)		
Initial	1.94	1.05
After RealEstate	2.49	1.01
After Lecture	3.26	0.91

A Student's t-test (two-tail) produced probabilities of 0.0057, 0.615, 0.005 for comparing *Initial to After Lecture*, *After Lecture to After RealEstate*, and *Initial to After RealEstate* respectively for inheritance. Similarly for iterators, the probabilities produced were $3x10^{-7}$, 0, $4x10^{-15}$ for comparing *Initial to After RealEstate*, *After RealEstate to After Lecture*, and *Initial to After Lecture* respectively.

TABLE 2
KNOWLEDGE ASSESSMENT (SCALE: 0-1)

	Mean	**Standard Dev.**
Inheritance (Conventional then Real Estate)		
Initial	0.50	0.23
After Lecture	0.56	0.24
After Real Estate	0.56	0.22
Iterators (RealEstate then Conventional)		
Initial	0.28	0.22
After RealEstate	0.40	0.24
After Lecture	0.50	0.22

III. Additional Survey Results

The final survey was designed to gauge some of other possible factors that may have influenced this study as well as give students opportunity to offer free-form comments.

The results of this survey support our choice that inheritance and iterators were acceptable topic choices for this experiment. Students had comparable knowledge about both topics and comparable interest in them. While the difficulty may be of slightly different nature (inheritance being largely a topic with much more breadth, while iterators being one difficult largely for its abstract nature) the two topics do appear reasonably alike to evaluate our results solely on the treatment results.

Additionally, this follow-up survey allowed for free-form comments from students. While there were select comments either strongly praising RealEstate or strongly opposing it, most comments focused on liking different aspects of both lectures and RealEstate. Some select comments (minor mistakes corrected):

- "The lab exercises significantly helped my understanding of both of the topics. I think. The lectures also helped, but by using the program in the lab helped it much more."
- "Actually doing something helps me better rather than just sitting there and listening. Interactive!"
- "I found the lecture-based instruction more helpful, but that may have been because it was done first. Since I understood it, RealEstate seemed a little redundant."
- "The lab exercises significantly helped my understanding of both of the topics. I think. The lectures also helped, but by using the program in the lab it helped much more."
- "The class lectures were great… but the RealEstate program was a bit confusing… I don't even know how it relates to iterators or inheritance."
- I thought the lecture and slides were pretty helpful, and I thought the lab was cool. Coding is the best way for me to learn concepts, so I really enjoyed having to work with code to learn the concepts of inheritance and iterators."
- "I feel as if I learn best with a combination of the two. Only one or the other doesn't give me my full understanding. I like things explained, to see examples and then to implement on my own in a lab setting."
- "I equally got the same value from the lectures and lab. They each had their own benefits."
- "I thought the RealEstate program really helped elucidate a lot of the mystery behind inheritance."
- "I think, for me at least, both educational methods are useful and necessary."
- "Lecture is the best way to learn the basic. The lab is the best way to learn 'what if' questions from the basic."
- "Combination of lecture and lab is useful, in which lecture teaches the concept, and lab puts the concept into practice."

TABLE 3
ADDITIONAL SURVEY

Question (5 – very, 1 – not at all)	Mean	Standad Dev.
1. How well did you understand inheritance before taking this class?	2.19	1.29
2. How well did you understand iterators before taking this class?	2.05	1.32
3. How interesting do you find inheritance?	3.16	1.09
4. How interesting do you find iterators?	2.99	1.12
5. How accessible did you find the lectures about inheritance and iterators?	3.83	0.94
6. How accessible did you find the lab involving the RealEstate program?	3.39	1.10
7. How useful did you find the RealEstate activity in your learning?	3.19	0.98
8. How useful did you find the lectures about inheritance and iterators in your learning?	3.90	0.93

CONCLUSIONS

From this data it is possible to gather that both methods do work reasonably well and are helpful – although there are particular subtleties worth noting. It is also relevant to note that from the last survey result, students expressed a roughly similar knowledge (and interest) of both inheritance and iterators. While the difficulty may be of slightly different nature (inheritance being largely a topic with much more breadth, while iterators being one difficult largely for its abstract nature) the two topics do appear reasonably alike to be able to focus just on the treatment results.

Judging by the summary survey data it seems that no matter what order the treatments were given, the lecture had a significantly greater impact (approximately double) on self-confidence than RealEstate. This may be explained by students being more familiar with this type of instruction from other classes the students have taken to and by the easier access difficulty (as shown by the last survey results). Moreover, it can also be seen that both treatments did have significant positive impact on self-perceived confidence. The

free form comments do mention how both the lecture and the lab activity were useful learning tools. This is also seen given that the overall means did not go down after any treatment.

However, it is worthwhile to look at the change made by RealEstate as the second treatment (for inheritance). Looking at the overall knowledge assessment it seems it made non-significant improvement (despite improving self-confidence). Thus, in addition with comments, it seems like follow-up of a software program only helps students reinforce their concepts rather than potentially establish new understanding. While some of this reinforcement may be positive and help student with later retention (something not addressed in this study), it seems like this is a relatively small gain. Conversely, showing a larger software artifact in the initial points of the learning may produce a much more significant gain on understanding compared to lecture.

Overall it seems that while both methods are effective in some ways, there are numerous subtleties that need to be considered before their usage. A larger software artifact is primarily useful as a self-confidence reassurance tool after the topic has been covered by a lecture – not as a tool to get some more students to learn that were not engaged through the lecture. However, using a larger software tool can potentially help students see different perspectives of a topic when a well-selected software program is picked and some guidance is offered about its perusal.

Additional investigation into how these large artifacts affect long-term retention (or ability to quickly re-acquire the knowledge) may be a beneficial study.

ACKNOWLEDGMENT

We are extremely grateful to the developers of the RealEstate software at North Carolina State University, Laurie Williams, Sarah Smith Heckman, and Dright Ho, for creating this artifact and making it available for use by educators.

REFERENCES

[1] Chickering, A. and Gamson, Z. Seven principles of good practice in undergraduate education. *AAHE Bulletin*, 39, pp. 3-7, 1987.

[2] Prey, J. C., Cohoon, J. P., and Fife, G. "Software Engineering Beginning in the First Computer Science Course." In the *Proceedings of the 7th SEI Conference on Software Engineering Education*, 1994.

[3] Tomayko, James E. "Teaching Maintenance Using Large Software Artifacts." In *Proceedings of the SEI Conference on Software Engineering Education*, 1989.

[4] Meneely, A., Williams, L., Gehringer, E. F., "ROSE: a repository of education-friendly open-source projects", in *Proc. of the 13th Annual Conference on Innovation and Technology in Computer Science Education*, pp.7-11, Madrid, Spain.

AUTHOR INFORMATION

Oleg Krogius, Graduate Student, Department of Computer Science, krogius@umich.edu.

Thomas B. Horton, Associate Professor, Department of Computer Science, University of Virginia, horton@cs.virginia.edu

Mark S. Sherriff, Assistant Professor, Department of Computer Science, University of Virginia, sherriff@cs.virginia.edu