Analogies in an Upper Division Operating Systems Course

Dr. Maria R. Ebling, United States Military Academy

Maria R. Ebling is an assistant professor at the United States Military Academy at West Point in the Department of Electrical Engineering and Computer Science. She earned both a Ph.D. and M.S. degree in Computer Science from Carnegie Mellon University and a B.S. in Mathematics from Harvey Mudd College. Her research interest includes pervasive computing, the Internet of Things, and Computer Science education. Prior to joining the faculty of West Point, Dr. Ebling spent three years as the Chief Technology Officer at Medaptive Health and nearly 20 years at the IBM T. J. Watson Research Center. Dr. Ebling is an ACM Distinguished Scientist.

Dr. Ryan Edward Dougherty, United States Military Academy

Dr. Ryan Dougherty is an assistant professor at the United States Military Academy at West Point. He earned his Ph.D. and B.S. degrees in Computer Science from Arizona State University. His research interests include theoretical Computer Science, evolutionary computing, combinatorial algorithms and Computer Science education. Dr. Dougherty spent a year as a visiting assistant professor at Colgate University. He has also been on several program committees in Computer Science education.

Dr. Nicholas Clark, University of St. Thomas

Nicholas Clark is an Associate Professor at the University of St. Thomas in Minnesota in the Department of Mathematics. He earned a Ph.D. in Statistics from Iowa State University and a B.S. in Mathematics from the United States Military Academy. His research interests include spatio-temporal statistical modeling and statistical pedagogy. He is also an associate editor for the Spatial Statistics Journal as well as Military Operations Research Journal.

Analogies in an Upper Division Operating Systems Course

Abstract

Analogies have a long, and sometimes controversial, history of use in teaching. The question of whether instructor-provided analogies or student-generated ones are more effective for learning in computer science has not been answered. In this study, we examined three different analogy treatments: no analogy, an analogy provided by the instructor, and analogies generated by students during an in-class exercise. We applied these treatments over two offerings of the Operating Systems (OS) course at our institution. We found no statistically significant results across these three treatments. Sometimes students who received no analogy performed better and at other times students who received either the student-generated analogies treatment or the instructor-provided analogies treatment performed better. We also compared the use of either analogy treatment to the use of no analogy, again finding no statistically significant differences. We conclude that instructors can select the teaching method most applicable to their lesson. If an active learning component is needed, then selecting the student-generated analogies option was found, anecdotally, to generate high student engagement. If time is short, the analogy can be skipped altogether.

Introduction

People tend to learn new concepts by connecting them to those they already know. This simple observation dates back to at least 1890 and to William James, the father of modern psychology. Analogical learning leverages this observation helping students draw concrete analogies between the topic they are trying to learn and one they already know.

Many people have studied the use of analogies in teaching computer science, predominantly in introductory computer science courses. Few studies have looked at whether instructor-provided analogies or those generated by students result in better learning outcomes, and few studies have looked at upper-division courses. Curious about the best instructional approach, we designed and executed an experiment to investigate the following questions:

- **RQ1**: Which type of analogy yields higher performance in an upper-division computer science course: student-generated or instructor-provided?
- **RQ2**: Do analogies, whether student-generated or instructor-provided, offer benefit over not using an analogy for learning upper-division computer science course material?

- **RQ3**: Does a student's overall performance in the computer science major influence the success of analogy usage?

In this paper, we share the results of a study, approved by our local institutional review board, that evaluates the use of instructor-provided and student-generated analogies, comparing them both to the case where no analogies are used. The paper is organized as follows. We begin with a discussion of related work and then summarize our study method. We then summarize the results of the study, discuss the results and our interpretation of them. Following that, we consider the limitations of this study and threats to its validity. We close with a discussion of future work, and some final thoughts to conclude the paper.

Background

Analogies have been applied to learning in computer science for decades. Gentner [1] viewed an analogy as a mapping from a base domain, the one already understood by the student, to the target domain, the one under study. Fincher and her colleagues [2] report the results of an ITiCSE working group on *notional machines* and have expanded the concept of the notional machine from that of Du Boulay's definition ("the general properties of a machine that one is learning to control" that is used by students learning to program) to include more general analogies. Sorva explains that "the purpose of a notional machine is to explain program execution" [3]. The ITiCSE working group identified 43 notional machines, which include three categories: machine-generated representations, handmade representations, and analogies. Their definition of analogies is consistent with that of Gentner's.

Macfarlane and Mynatt [4] studied the use of advanced organizers (aka analogies) in teaching computer programming concepts. They found only small differences in learning that required only syntactic knowledge, but a significant improvement in learning that required semantic knowledge with the use of an analogy.

The use of analogies in introductory computer science has been studied extensively. Duvall [5] advocates for using stories, many of which contain analogies, in teaching computer science and provides numerous examples of stories relevant to computer science courses. Similarly, Forišek and Steinová [6] describe numerous analogies for use in an algorithms course. Neither Duvall nor Forišek and Steinová formally evaluated their analogies in a classroom setting.

Sanford et al. [7] studied analogies used in university-level, introductory computer science education and differentiated between atomic analogies that map a single feature between the source and target domain and complex analogies that map multiple features. In addition to limiting their study to introductory-level courses, they also did not examine the source of the analogy nor did they assess the success of learning when analogies were used.

Schez-Sobrino et al. [8] studied the use of a road sign analogy in teaching introductory computer science. They then described how that analogy could be extended to apply to teaching the concepts of concurrency and agent-based programming. Although they evaluated how well students in their introductory course understood the components of the road-based analogy for basic programming concepts, they did not evaluate the student understanding of the more complex concepts.

Hermans et al. [9] studied the use of analogies (which they called metaphors) in teaching novices, both children and adults, how to program. They compared the learning outcomes achieved when using two different analogies, but only considered instructor-provided ones. Their study compared two alternative instructor-provided analogies appropriate to a single lesson in a pre-college introduction to programming course.

Some studies have also examined analogies generated by students. Tartaro et al. [10] studied whether student-generated content impacted learning outcomes and showed that analogies, whether instructor-supplied or student-generated, improved performance on exams in a bioorganic chemistry course. Bettin et al. [11] studied analogies in a concurrent programming course, which, like our course, is a junior- or senior-level course at their institution. The instructors used analogies in their teaching and their study asked survey questions to understand what analogies students generated in completing assignments. Their study provided rich anecdotal data, though they did not measure learning outcomes.

Finally, Frigg and Hartmann discuss the extensive history of philosophical debates about the idealization of models [12]. They describe two types of idealizations. The *Aristotelian* idealization strips away unnecessary detail to support understanding of the basic idea. This type does not lie — it simply remains silent on certain aspects of the truth. The *Galilean* idealization distorts the true nature of the thing being modeled to make it easier to understand. Similarly, analogies can be either Aristotelian or Galilean. The instructor-provided analogies and the student-generated analogies in our study were predominantly Galilean, but a few were Aristotelian.

In contrast to prior work, our study examines learning outcomes associated with the use of analogies in an upper-division undergraduate course on operating systems. We consider both student-generated analogies and instructor-provided analogies and compare those to the lack of an analogy.

Method

Our study used three different treatments: no analogy, instructor-provided analogy, and student-generated analogy. We describe each, including a summary of the intructor-provided analogies and a summary of how we implemented the student-generated treatment. We also describe how the learning outcomes were assessed and the data that we collected.

Treatments

We performed this study in the context of a senior-level, undergraduate course in Operating Systems (OS) over a period of two semesters: Fall 2022 and Fall 2023. Each course had three sections. We used three lesson topics that had a natural analogy component. We applied three teaching treatments: no analogy, student-generated analogy, and instructor-provided analogy. For each of three lessons, each section was assigned a different treatment and we rotated the treatments around the three sections of each course across the three lessons, such that each student experienced each treatment once and such that each topic was taught using each treatment once.

No Analogy

The “no analogy” treatment served as a control. In this treatment, we presented information about the topic and then presented a summary without using an analogy. We generally used one or two slides of information containing only words, without any images. These slides cover the same material presented to students prior to the introduction of these analogies.

Student-Generated Analogy

In the “student-generated analogy” treatments, the instructor presented students with a summary of the topic and then asked students to come up with an “everyday” analogy related to the topic. After giving students several minutes to design their own analogy, they were then asked to respond to a short survey in which they described their analogy. At this point, students were asked to share their analogy with a partner (in the case of an odd number of students with two partners). The pair (or triple) were then asked to select the best analogy and consider ways the analogy could be expanded and think about where the analogy breaks down. They then documented this discussion in a short survey as well, including whom they partnered with.

Instructor-Provided Analogy

For the “instructor-provided analogy” treatment, the instructor presented information about the topic, and then provided their own analogy. The instructor presented the analogy using a visually rich set of slides. The three analogies provided by the instructor are summarized below.

Processes vs. Threads: Processes and threads are like an apartment building. Each apartment is like a process and each person who lives in that apartment is like a thread. People share physical space and threads share an address space. People share the kitchen and laundry room just as threads share code, files, and data. Certain resources, like a bathroom, require synchronization because only one person can use it at a time, like a critical section. Each person has personal items, like a bed and a dresser, just like each thread has its own stack. The building owner can evict existing tenants and assign new ones, just like the operating system can kill existing processes and create new ones. (This analogy is Aristotelian in that it leaves out important details such as the process control block and it ignores scheduling.)

Virtual Memory: Virtual memory swapping is like a convenience store. Convenience stores sell the products you use all the time just as virtual memory keeps the data needed by the processes easily at hand. Convenience stores do not have much shelf space, just as computers do not have much physical memory. Both rely on locality of reference. Convenience stores shelve the popular items and those shelves have to be restocked frequently whereas virtual memory keeps the most popular pages in physical memory and reloads them as needed. Convenience stores use a storage room to store extra supplies and restocking from the storage room takes time; virtual memory uses the disk to store extra pages from memory and swapping these pages to/from disk also takes time. Convenience stores keep an inventory so that they know where all the products are stored and shelved, likewise virtual memory keeps a page table that tracks where each page is located and whether or not it is dirty. (This analogy is Galilean in that it distorts the frequency with which store shelves are restocked and the fact that pages of memory can be placed on *any* shelf.)

Table 1: Dates and Attendance for Lessons and Events

		Population (Major GPA)					
Term	**Topic**	**None**	**Student**	**Instructor**	**Lesson**	**Exam**	**Days**
Fall 23	Processes	9 (3.2)	14 (3.1)	12 (3.5)	Sept 7	Sept 26	19
	Virtual Memory	17 (3.1)	10 (3.4)	12 (3.3)	Oct 12	Nov 3	22
	File Systems	15 (3.4)	12 (3.3)	16 (3.1)	Nov 9	Dec 13	34
Fall 24	Processes	14 (3.3)	15 (3.5)	15 (3.2)	Oct 16	Dec 16	61
	Virtual Memory	13 (3.3)	15 (3.6)	14 (3.3)	Sep 29	Nov 15	47
	File Systems	16 (3.5)	13 (3.3)	14 (3.3)	Nov 28	Dec 16	18

File Systems: File systems are like libraries; they store knowledge for the long term. Card catalogs tell us where to find books, just like a FAT or an index block tells us where to find file segments. Library stacks provide organization much like directories. Some books reside in the "restricted section" just like some files have access restrictions. (This analogy is Galilean in that libraries are generally read-only whereas files are frequently read-write.)

Assessments

Each topic was covered on either a midterm or final exam to assess student learning. We use the students' grades on the relevant exam question to assess their understanding of the specific topic.

Data Collection

Table 1 documents the dates of the lessons in which the analogy topics were covered. For each topic, it also documents the date of the exam used to measure student learning and the number of days between the analogy lesson and the exam. We also show the number of students in each treatment population and their average GPA in their major at the start of the course.

Data associated with any student who missed class, arrived late, or departed early on the day of the analogy lesson has been removed from our analysis. One student did not answer the question about file systems on the final exam. We removed this student from our analysis of the file system topic.

Results

In this section, we present the results of the study. We start with the statistical analysis of student learning. We then share a few of the analogies generated by students.

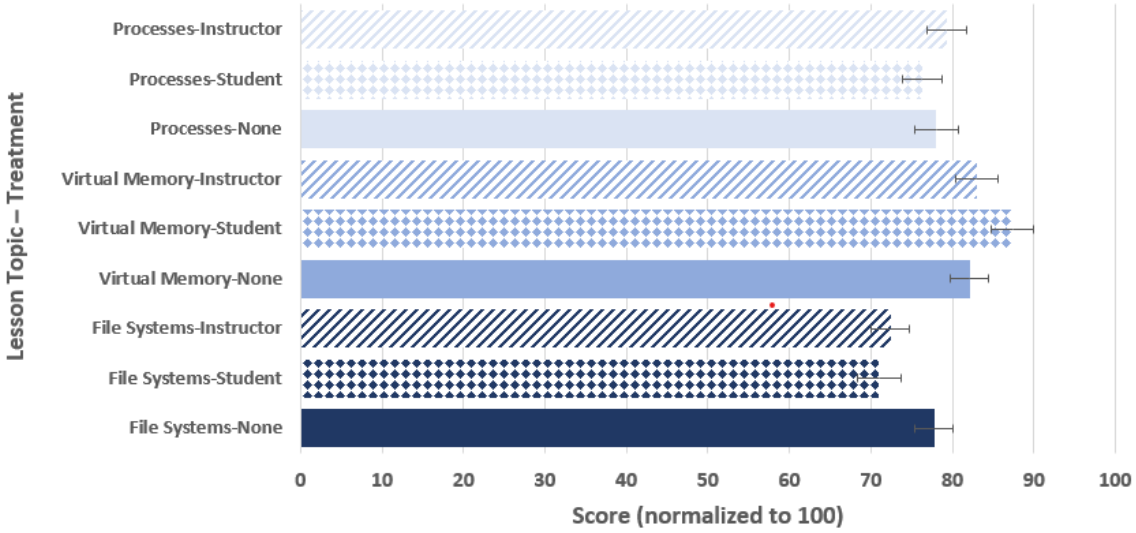
Statistical Analysis

We performed a mixed-effects multiple regression analysis treating student as a random effect to control for the multiple observations per student. The design was a randomized block design with

Table 2: Effects of Treatment, Topic, & Interaction

	Sum Sq	Mean Sq	F value	p(>F)
Treatment	9759	4879	0.4508	0.6379
Topic	445383	222692	20.5735	1.11e-08
Academic Year	536458	536458	49.5609	4.47e-10
Major GPA	731617	731617	67.5908	2.48e-12
Treatment:Topic	20883	5221	0.4823	0.7487

Figure 1: Assessment Scores Across Analogies



repeated measures, each class (block) was assigned a random starting treatment. Although the block effect is confounded with the treatment/topic interaction, this was found to be statistically insignificant. We considered a p-value of 0.05 to represent a statistically significant result. Treatment, topic, student major GPA, and academic year of the course were treated as fixed effects. We further fitted a model with an interaction between treatment and topic; however, this model was not preferable to the simpler model without the interaction term (p=0.74). This suggests that there is no evidence that analogies work different across different topics.

The treatment effect was not shown to be statistically significant (p=0.64), although topic ($p<0.001$), academic year the course was taken ($p<0.001$), and the student's major GPA ($p<0.001$) all had a statistically significant effect on the student's score.

Even though the interaction term was not significant, general trends in the significance of the treatment may yield potential threads for future research. To this end, we further examined the marginal effects of the treatment and topic combination to discern whether any notable data patterns emerged. (The marginal means analysis is a post-hoc test to determine where the differences are when the ANOVA comes up statistically insignificant.) The marginal means plots in Figure 1 show the confidence intervals for each topic-treatment combination. The different

Table 3: Use of Analogy or Not

	Sum Sq	**Mean Sq**	**F value**	**p(>F)**
Analogy (Yes/No)	9152	9152	0.8520	0.3574
Topic	89366	44683	4.1594	0.01678
Academic Year	537298	537298	50.0159	3.65e-10
Major GPA	750897	750897	69.8993	1.02-12
Treatment:Topic	8673	4337	0.404	0.6683

shading indicates the topic: light blue for processes, medium blue for virtual memory, and dark blue for file systems. The texture indicates the treatment (diagonal lines for instructor-provided, dots for student-generated, and solid for no analogy). Here we observe no discernible patterns with the average for no analogies sometimes higher than the other two categories as in file systems, and other times student-generated analogies had a higher average as in virtual memory. However, the confidence intervals within a given topic significantly overlap, suggesting no treatment effect for a given topic.

We conclude that the three treatments do not lead to statistically significant differences in learning outcomes. Further, the direction of impact of the effect is not consistent, meaning sometimes having no analogies performed the best and other times instructor-provided analogies performed the best.

Table 2 shows the ANOVA[1] for the results which again shows effects due to topic, academic year, and major GPA but no effect due to treatment or the interaction between treatment and topic. We then combine the results from both the student-generated and the instructor-provided analogies to understand the impact *any* analogy had on learning. Once again, sometimes the analogy treatment outperforms the no analogy treatment and sometimes not. We observe that none of the results were statistically significant (p=0.36) and have omitted the figure. Table 3 shows the ANOVA for the results, which again show effects due to the topic, academic year, and GPA but not to the use of an analogy or the interaction between treatment and topic.

Next we examined the performance of students in the top, middle, and bottom third of their class based on their major GPA at the start of the course. We ran the mixed effects regression separately for each group of students. The effects of the treatment on the bottom third of students were similar to the entire body as seen in Table 4. Here we see no effect due to treatment, but an effect due to topic and academic year.

Amongst the middle third of students, the treatment had an effect and that effect changed depending on the topic as seen in Table 5. Although this finding is potentially interesting, most of the significance occurred due to the poor performance of this group on the student analogies for the file systems topic where the average score was 470 (95% CI of 354 - 586). Due to the random dispersion of students across the classes, only four students out of the 23 middle third received this treatment-topic combination, which suggests this may be a random outcome due to the small sample rather than a true finding.

[1] All are Type 3 ANOVA with Satterthwaite's Correction.

Table 4: Effect of Treatment on Bottom Third

	Sum Sq	Mean Sq	F value	p(>F)
Treatment	11452	5726	0.6007	0.55244
Topic	73430	36715	3.8518	0.02802
Academic Year	210143	210143	22.0461	6.481e-05
Treatment:Topic	18536	4634	0.4861	0.74579

Table 5: Effect of Treatment on Middle Third

	Sum Sq	Mean Sq	F value	p(>F)
Treatment	142761	71381	7.0465	0.00210
Topic	354622	177311	17.5037	1.978e-06
Academic Year	77194	77194	7.6204	0.01015
Treatment:Topic	209111	52278	5.1607	0.00164

Amongst the top third of students, the treatment had a moderate effect on student performance (p=0.098); see Table 6. Interestingly, these students performed worse with the instructor-provided analogies than either the student-generated ones or none for each of the three topic areas. The marginal mean for instructor-provided was 828 (95% CI of 788 - 868), whereas the marginal mean for student-generated was 879 (839-919) and the marginal mean for none was 881 (838 - 924).

Over the course of our experiment, we made two anecdotal observations. First, both instructors noted high levels of student engagement with the student-generated analogies exercises. Second, the students generated a diverse set of analogies.

Student-Generated Analogies

We next provide some examples of the student-generated analogies.

Processes:

- Processes are like a beehive and threads are like bees. The queen bee is the operating system. The bees share resources, like honey and pollen. Worker bees dance to communicate information about where to find pollen. (This analogy is Galilean because the queen bee is not directing the worker bees nor giving them turns at completing their work.)

Table 6: Effect of Treatment on Top Third

	Sum Sq	Mean Sq	F value	p(>F)
Treatment	44952	22476	2.4305	0.098481
Topic	183030	91515	9.8962	0.000244
Academic Year	57334	57334	6.1999	0.019576
Treatment:Topic	15367	3842	0.4154	0.796580

- Processes are like a team leader and threads are like their subordinates. The team leader could do everything, but that takes a significant amount of time and the work is centralized in one person. Instead, if the team leader assigns work items to subordinates, people can work concurrently and thus save time. Subordinates can run into errors on their task. The students noted that the analogy breaks down if a subordinate gets blocked on a task because their peers or their team leader would be there to help them out to ensure the tasks get completed. (This analogy is Galilean because the subordinates are truly operating in parallel and not sharing a finite set of central processing units.)

Virtual Memory Swapping:

- Virtual memory is like storing possessions in a dormitory. The dressers and shelves are the most accessible, followed by under-bed storage, followed by the trunk room. The items frequently used are stored in the dresser or on the shelf. Items less frequently used are stored under the bed. Items rarely used are stored in the trunk room. Items can be moved between these storage locations as their frequency of use changes. (This analogy is Galilean because students do not typically move items around the different storage areas as regularly or as methodically as virtual memory.)
- Virtual memory is like having a refrigerator in the kitchen and another in the garage. The most commonly used items are stored in the kitchen fridge whereas the least common used items are stored in the garage. (This analogy is Galilean because people do not typically move items between two fridges as regularly and methodically as virtual memory.)

File Systems:

- A file system is like a city or map. All roads lead to Rome (the root). A folder or directory corresponds to a neighborhood; a file corresponds to a house. Symbolic links that connect different parts of the file system are like tunnels between cities. (This analogy is Aristotelian because it leaves out many details, such as mounting file systems, as noted by the students.)
- A file system is like a tree. It starts at the root and has branches. Each branch can break into more branches. These branches are like directories. The leaves on each branch are like files. (This analogy is Aristotelian because it leaves out many details.)

Discussion

In this section, we examine the results in light of each research question and we highlight some anecdotal observations.

RQ1: Instructor vs. Student Analogies

We first consider the question of whether instructor-provided analogies or student-generated ones yield higher performance. Examining Figure 1 with this question in mind we see the mean scores across instructor-provided and student-generated analogies. The instructor-provided scores are shown with diagonal lines and the student-generated scores are shown with dots. We observe that instructor-provided analogies appear to score higher than student-generated analogies for two

topics (processes and file systems), but not for the third (virtual memory). We also observe that the figure clearly shows overlapping confidence windows. We found no significant differences between these two treatments (Table 2).

We also considered whether the instructor-provided analogies might have been poorly chosen. To examine this question we compared student performance year-over-year to see if one of the instructor-provided analogies performed consistently poorly. In all cases, student performance under the different treatments from one year to the next had no statistical significance.

RQ2: Analogies or Not

We next consider the question of whether analogies, either instructor-provided or student-generated, offer benefit over no analogy. Again, referring to Figure 1, focus on only the differences between the no analogy treatment (shown in the solid bars) and the two analogy-based treatments (shown in the bars containing lines or dots). Students appear to perform worse if they had an analogy for file systems, but slightly better with an analogy for virtual memory; the result for processes was mixed. Again we observe overlapping confidence windows. We found no statistically significant differences based on the presence or absence of an analogy treatment (Table 3).

RQ3: Analogies for Strong/Weak Students

We analyzed the performance of students in the top, middle, and bottom third of their class based on their major GPA at the start of the course (Tables 4-6). The only statistically significant result was seen in the middle third, where students who received the student-generated analogies on the file system topic performed significantly worse than those who received either no analogy or the instructor-provided analogy. We note that only four students fell into this group, and conclude that it may be due to the small sample size more than a true finding.

Anecdotal Observations

In our experiment we made three anecdotal observations. The first relates to student engagement. Both instructors noted high levels of student engagement during the analogy discussion. The second is that both instructors noted a diverse set of analogies, including many we had never considered. Finally, the analogies rarely appeared in written responses to open-ended questions in our assessments.

Two students mentioned an analogy in their responses to an open-ended question on an exam. One student referenced a student-generated processes analogy in the first mid-term and the other student mentioned the instructor-provided, convenience store analogy in the second mid-term in the Fall 2022. In both cases, the students who mentioned the analogy were among the lower-performing students. This observation leads to additional questions, such as whether there is any significance to the fact that two lower-performing students mentioned analogies on exams.

Limitations & Threats to Validity

This study used a sample of convenience, with participants drawn from the students enrolled in our OS courses during the Fall of 2022 and 2023. The demographics of the participants from these two classes are 9% female and 91% male; 25% Asian, 10% Black or African American, 17% Hispanic, 1% Multi-Ethnic, 43% White, and 3% international. Note that we found no statistically significant differences in performance based on treatment used related to either gender or race; details omitted due to space.

This study examined the use of analogies in an upper-division Operating Systems course, taught by two instructors at a single institution. Approximately fifty students take this course each year, during the fall semester. With such small numbers, achieving statistical significance is challenging. In addition, the number of participants in some lessons was much fewer than expected due to a virus that was circulating amongst the student body that resulted in an unusually high number of absences (especially during Fall 2022), further complicating our ability to achieve statistical significance.

The instructor-provided analogies have the advantage of better visuals. For each analogy, the instructor created a set of slides to illustrate the base domain and relate it to the target domain. These visuals may have unintentionally increased the instructor-provided analogy scores. Future studies may want to forego the use of visual illustrations when comparing performance to student-generated analogies when the student-generated analogies do not give sufficient time to the exercise to generate visual supports. We note that, despite the advanced visuals offered by the instructor-provided analogies in OS, the instructor-provided analogies did not seem to provide an advantage.

This student-generated analogies have the advantage of more time spent on the task. For each analogy, the students were given time in class to identify real-life analogies to the topic. This resulted in this treatment modality giving the students more time. We note that, despite the additional time spent on the task, the student-generated analogies did not seem to provide an advantage.

Students may no longer be able to relate to the analogy used for the file systems topic because libraries have changed so much in recent years. It is possible this analogy is not helpful.

Many students study in groups outside of class. It is certainly possible that they shared the instructor-provided analogies without realizing the impact that would have on this study. Likewise, it is possible that students exposed to the student-generated analogies might have continued to create analogies on other topics for the remainder of the course.

We did find statistical significance with respect to retention. The relationship was inversely related, meaning the students performed better when there was more time between the lesson and the exam. This result is explained fairly well by Roediger and his colleagues [13]. It turns out students also performed better on processes than file systems and, as seen in Table 1, the exam in which the process topic was assessed was more than three times as many days after the lesson as was the exam that assessed file systems. Thus, it is impossible to disentangle the retention effect from that of the topic. The solution to this issue would be to randomize the length of time between the lesson and the topic, but that is not realistic.

Future Work

One possible area for future work would be to expand the study to a larger number of students across a broader set of courses, including introductory and upper division courses. This expansion would allow us to increase the diversity of the teaching material and instructors, increasing our confidence in the answers. However, to obtain a definitive answer to these questions, we need a larger community than is available at our institution. Such an expansion would greatly increase the diversity of teaching material, instructors, and students and help to provide the community with a definitive answer to these questions.

The analogies created by the OS students offer a rich source of data. We plan to examine these analogies to better understand their characteristics. Duran and his colleagues [14] also considered criteria for evaluating models of program behavior that we plan to leverage.

Another possible area for future investigation is examining a slightly different treatment for the student-generated analogies by giving students a starting point, one that is known to lead to a good analogy. This approach allows students to do the thinking and analysis of the student-generated analogy, but does not require them to come up with the initial connection.

Conclusions

This study provides initial answers to three questions. The first question asked which type of analogy, those provided by the instructor or those generated by students, yields higher performance in an upper-division computer science course. The second question asked whether analogies provide benefit to students of upper-division computer science courses. The final question asked whether a student's overall performance in the computer science major predicts the value of either the use of analogies or the type of analogy to their learning. Our findings to all of these questions provide no statistically significant results, leading us to the conclusion that instructors should select the one that best matches the needs for their lesson. If they need an active learning component, the student-generated analogies were found to provide a high level of student engagement. If they find themselves short on time, they can skip the analogy altogether. Their choice neither harms nor helps the students.

Acknowledgments

We thank the student participants from both the Fall 2022 and the Fall 2023 offerings of CS481 for their willingness to support computer science education research. The views expressed in this article are those of the author(s) and do not reflect the official policy or position of the Department of the Army, Department of Defense, or the U.S. Government.

References

[1] D. Gentner. *The Structure of Analogical Models in Science*. 4451. Available at https://archive.org/details/DTIC_ADA087625/page/n1/mode/2up. Bolt Berenek and Newman, Inc., July 1980.

[2] S. Fincher et al. “Capturing and Characterising Notional Machines”. In: *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. ITiCSE ’20. Trondheim, Norway: Association for Computing Machinery, 2020, pp. 502–503. ISBN: 9781450368742. DOI: 10.1145/3341525.3394988. URL: https://doi.org/10.1145/3341525.3394988.

[3] J. Sorva. “Notional machines and introductory programming education”. In: *ACM Trans. Comput. Educ.* 13.2 (July 2013). DOI: 10.1145/2483710.2483713. URL: https://doi.org/10.1145/2483710.2483713.

[4] K. N. Macfarlane and B. T. Mynatt. “A Study of an Advance Organizer as a Technique for Teaching Computer Programming Concepts”. In: *Proceedings of the Nineteenth SIGCSE Technical Symposium on Computer Science Education*. SIGCSE ’88. Atlanta, Georgia, USA: Association for Computing Machinery, 1988, pp. 240–243. ISBN: 089791256X. DOI: 10.1145/52964.53024. URL: https://doi.org/10.1145/52964.53024.

[5] S. Duvall. “Computer Science Fairy Tales”. In: *J. Comput. Sci. Coll.* 24.2 (Dec. 2008), pp. 98–104. ISSN: 1937-4771.

[6] M. Forišek and M. Steinová. “Metaphors and Analogies for Teaching Algorithms”. In: *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*. SIGCSE ’12. Raleigh, North Carolina, USA: Association for Computing Machinery, 2012, pp. 15–20. ISBN: 9781450310987. DOI: 10.1145/2157136.2157147. URL: https://doi.org/10.1145/2157136.2157147.

[7] J. P. Sanford et al. “Metaphors We Teach By”. In: *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*. SIGCSE ’14. Atlanta, Georgia, USA: Association for Computing Machinery, 2014, pp. 585–590. ISBN: 9781450326056. DOI: 10.1145/2538862.2538945. URL: https://doi.org/10.1145/2538862.2538945.

[8] S. Schez-Sobrino et al. “ANGELA: A Novel Approach of Graphic Notation Based on the Metaphor of Road Signs to Facilitate the Learning of Programming”. In: *Proceedings of the Seventh International Conference on Technological Ecosystems for Enhancing Multiculturality*. TEEM’19. ACMsearch1. León, Spain: Association for Computing Machinery, 2019, pp. 822–829. ISBN: 9781450371919. DOI: 10.1145/3362789.3362871. URL: https://doi.org/10.1145/3362789.3362871.

[9] F. Hermans et al. “Thinking out of the Box: Comparing Metaphors for Variables in Programming Education”. In: *Proceedings of the 13th Workshop in Primary and Secondary Computing Education*. WiPSCE ’18. Potsdam, Germany: Association for Computing Machinery, 2018. ISBN: 9781450365888. DOI: 10.1145/3265757.3265765. URL: https://doi.org/10.1145/3265757.3265765.

[10] A. Tartaro et al. “Learning Outcomes From a Student-Generated”. In: *Proceedings of the 19th International Conference on Supporting Group Work*. GROUP ’16. Sanibel Island, Florida, USA: Association for Computing Machinery, 2016, pp. 449–452. ISBN: 9781450342766. DOI: 10.1145/2957276.2996291. URL: https://doi.org/10.1145/2957276.2996291.

[11] B. Bettin, L. Ott, and J. Hiebel. “Semaphore or Metaphor? Exploring Concurrent Students’ Conceptions of and with Analogy”. In: *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1*. ITiCSE ’22. Dublin,

Ireland: Association for Computing Machinery, 2022, pp. 200–206. ISBN: 9781450392013. DOI: 10.1145/3502718.3524796. URL: https://doi.org/10.1145/3502718.3524796.

[12] R. Frigg and S. Hartmann. "Models in Science (2nd edition)". In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta. Stanford, CA: Stanford, 2021.

[13] I. Henry L. Roediger and J. D. Karpicke. "The Power of Testing Memory: Basic Research and Implications for Educational Practice". In: *Perspectives on Psychological Science* 1.3 (2006), pp. 181–210. DOI: 10.1111/j.1745-6916.2006.00012.x. URL: https://doi.org/10.1111/j.1745-6916.2006.00012.x.

[14] R. Duran, J. Sorva, and O. Seppälä. "Rules of Program Behavior". In: *ACM Trans. Comput. Educ.* 21.4 (Nov. 2021). DOI: 10.1145/3469128. URL: https://doi.org/10.1145/3469128.