6-2023

More (Sema|Meta)phors: Additional Perspectives on Analogy Use from Concurrent Programming Students

Briana Christina Bettin
Michigan Technological University, bcbettin@mtu.edu

Linda Ott
Michigan Technological University, linda@mtu.edu

Julia Hiebel
Michigan Technological University, jshiebel@mtu.edu

Recommended Citation

Bettin, B. C., Ott, L., & Hiebel, J. (2023). More (Sema|Meta)phors: Additional Perspectives on Analogy Use from Concurrent Programming Students. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE, 1*, 166-172. http://doi.org/10.1145/3587102.3588831
Retrieved from: https://digitalcommons.mtu.edu/michigantech-p2/43

Follow this and additional works at: https://digitalcommons.mtu.edu/michigantech-p2

 Part of the Computer Sciences Commons

More (Sema|Meta)phors: Additional Perspectives on Analogy Use from Concurrent Programming Students

Briana Bettin
Michigan Technological University
Houghton, MI, USA
bcbettin@mtu.edu

Linda Ott
Michigan Technological University
Houghton, MI, USA
linda@mtu.edu

Julia Hiebel
Michigan Technological University
Houghton, MI, USA
jshiebel@mtu.edu

ABSTRACT

A concurrent computing course is filled with challenges for upper-level programming students. Understanding concurrency provides deeper insight into many modern computing and programming language behaviors, but the subject matter can be difficult even for relatively proficient students. It can be a challenge to help students navigate and understand these unfamiliar topics. While there is a difference in general programming familiarity, teaching this novel material is not unlike some challenges faced when engaging introductory students with first programming concepts.

In this work, we explore the use of analogy by students while learning a novel programming methodology. We investigate perceptions of the utility of analogy and creation of analogies in the concurrent course. We also examine perceptions of analogy value across students' computing education and factors which impacted their use or disuse of provided or student-generated analogies.

This exploration suggests that pedagogical analogy design can be memorable and significant for student understanding. It further suggests that analogies inherent in concept naming and foundational examples may have even greater salience. While not all students create analogies, those that do share both unique examples and additions to existing examples that helped them understand core concepts. Students had mixed responses on whether analogy as a tool was used in their lower-level courses. Despite this, most found analogies to be useful, with a majority finding them even more useful in upper-level programming courses.

CCS CONCEPTS

• **Social and professional topics** → **Computing education**; • **Computing methodologies** → *Concurrent computing methodologies.*

KEYWORDS

computing education, mental models, understanding, analogical reasoning, analogy, metaphor, concurrent computing, concurrent algorithms, qualitative

ACM Reference Format:
Briana Bettin, Linda Ott, and Julia Hiebel. 2023. More (Sema|Meta)phors: Additional Perspectives on Analogy Use from Concurrent Programming Students. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2023), July 8–12, 2023, Turku, Finland.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3587102.3588831

ITiCSE 2023, July 8–12, 2023, Turku, Finland

ACM ISBN 979-8-4007-0138-2/23/07.
https://doi.org/10.1145/3587102.3588831

1 INTRODUCTION

Concurrent programming is a powerful concept, one that many students struggle to incorporate into their existing understanding of computing. This difficulty can be magnified by the sense of competence or even expertise that upper-level students have often developed prior to encountering this starkly novel approach.

Given that analogy can be effective in bridging between known experiences and stories to unknown ideas, this seemed a useful tool to explore more deeply in concurrent curricula. As concurrent students are upper-level students, we were also curious about their perspectives on the use of analogy throughout their prior computing education. We center these research questions:

- What analogies do students remember and engage with across their time in a concurrent computing course? Across computing curricula?
- How do students characterize engagement with analogies when learning computing course material?
- How might student perspectives on analogies shift over their time in computing courses?

We identify patterns in concurrent computing students' perspectives on analogies through this exploration. These observations provide both insights and new paths for future research.

2 LITERATURE REVIEW

Analogy, as a pedagogical tool, has been employed in the teaching of concurrent and parallel computation. Anderson and Dahlin [1], a textbook often leveraged for such curricula, explores core topics such as race conditions, synchronization, and deadlock with analogical structures and examples. A parallel computing course described by Giacaman [15] uses an overarching analogy representing the system as a company, with desks representing processor cores and employees representing threads, and adds additional analogical elements as new concepts are presented. Many concurrent pedagogical environments also leverage visualization to explain and analogize program behavior to facilitate reasoning about the concurrent concepts [13, 20, 22]. Given the use of analogy within concurrent curricula, there is prospective value in exploring student perceptions and relationships with these pedagogical activities.

Shene and Carr [22] enumerate challenges in overcoming the sudden, significant paradigm shift students face when transitioning from a sequential to concurrent mindset. Common dynamic behavior bugs, such as race conditions and deadlock, can occur intermittently. This requires students to reason over their program

for correctness, as opposed to testing reliance. Further, they note: "most students indicate that they need additional and real examples in addition to the classical ones" [22]. "Classic" textbook examples are often limited to presenting a correct solution or examining an incorrect one in isolation. To maximize the value of "real eamples" (which depend on learner experiences and understanding), it becomes imperative to understand student interactions with analogy.

Recent developments in the state-of-the-art of teaching concurrent and parallel topics is given by Durraes et al [8]. They suggest that traditional pedagogical approaches still dominate the teaching of concurrency and advocate for additional research into the inclusion of modern pedagogical approaches to increase engagement and learning. Pedagogical engagement can be increased with analogy usage [2, 17, 19], further indication that analogy and concurrency are important topics for exploration.

In computing education research (CER), most work is focused on introductory programming courses and data structures [5, 7, 9, 12, 18, 21]. Usage in upper-level classes has been less explored in analogy work [4, 12, 15] across CER. There has been historical discourse opposing the use of analogy [11], which may have contributed to its under-exploration in the field. Recently, shifting conversations within CER have enumerated benefits of analogy [3, 16]. This shift in understanding is consistent with cognitive sciences and STEM education literature on analogy as a learning tool [3, 14].

3 OUR CONTEXT

Located in a rural area of the United States, our university student body is majority white males. Our concurrency course has a similar makeup, paralleling nationwide demographic trends for computer science enrollment. The course is required for computer science, software engineering, and computer engineering majors, and is typically taken by undergraduates in their third or fourth year. The required nature of the course means course demographics closely mirror university and department trends as well. Our Spring 2020 course had 81 students enrolled, with 67 completing the course.

Our concurrent computing course covers multi-process and multi-threaded programming concepts, with a focus on synchronization constructs, including mutual exclusion (mutex) locks, semaphores, monitors, and channels. Proper synchronization is required to prevent race conditions—incorrect results due to concurrent manipulation of shared resources, such as bugs which occur when a shared resource is manipulated between time-of-check and time-of-use (TOCTOU). The development of techniques for preventing race conditions extends beyond multi-process and multi-threaded programs, as sequential programs interact with shared operating system resources (most commonly the file system). Students are further introduced to the computer security vulnerability consequences of race conditions and improper synchronization. In order to support student understanding of the design and use of concurrency tools, relevant systems-level topics are discussed briefly.

4 METHODOLOGY

In the Spring 2020 Concurrent Course, a survey with the following six open-response questions was administered:

(1) In Project 4, you designed a Synchronization Protocol for a Landlord and Students sharing a room. What analogies did you construct surrounding mutexes and semaphores to understand the problem?
(2) In Project 5, you designed a Synchronization Protocol for Elves and Reindeer competing for the limited resource of Santa's attention. What analogies did you construct surrounding monitors and condition variables to understand the problem?
(3) In this course, several analogies have been used to explore the material and topics. Which of these have been most helpful for you in learning the material? Why?
(4) Have you or any study groups you are in developed any analogies throughout this course in an effort to learn the material? If so, what topic did the analogy entail, and what was it compared to?
(5) Have analogies been a tool you used in lower-level courses? What analogies helped you understand computing concepts in courses prior to this one?
(6) Has your perception or use of analogy changed as an upper-level student? Do you feel you use them more/less, or that they are more/less effective? What do you feel may have impacted any changes (or lack thereof) in your perception?

This survey was deemed "exempt" by the institutional ethics board. Extra credit was given to students for completing the survey with the requirement to receive credit being "thoughtful response of about 3-5 sentences per question". This ensured that effort was put into the responses, but that content of the responses did not impact credit given. Students were asked for their consent to use their responses in this research. Effort was the **only** requirement to receive the extra credit as indicated on the consent statement. A total of 55 students completed the survey and gave consent. These 55 responses were the basis for this research.

5 ANALYSIS

To analyze the qualitative data, survey responses were coded using key words and concepts, which were used to develop emergent themes. This aligns with a constructivist grounded theory [6] approach. While our approach fits well with this description, we do not wish to create method slur [23] by suggesting a complete constructivist grounded theory implementation. Specifically, we present observations of student experiences rather than propose a theory, which diverges from the intention of grounded theory. However, we align our approach with several key practices of grounded theory implementation, such as evolving research questions, coding developing categories, social construction of participant experiences, and sufficient original data for claims (82% of students completing the course gave permission for us to use their survey).

6 RESULTS

In this paper, we focus on Survey Questions 3 through 6, which are aimed at understanding students' general perceptions and use of analogy. Questions 1 and 2 were previously explored [4]. Through these four questions, our analysis focuses on their perception of pedagogical analogy use across their time both *as a concurrent student* and *as a computing major*. We identify here common themes which align with the research questions as well as potential interest to the broader research community.

6.1 Question 3 Response Themes

6.1.1 Dining Philosophers. The most commonly cited course analogy which students found helpful was the "Dining Philosophers" example, used to explain deadlock behavior through the imagery of several philosophers (threads) each wishing to eat (access shared resources), but requiring two chopsticks (mutex locks) when only one is set out for each of them. If each picks up a chopstick and waits, a circular dependency occurs as no one releases their chopstick, but no one has enough chopsticks to move forward (eat).

Students referenced this analogy as being quite memorable in content and presentation. They also indicated that they *reused* the analogy by revisiting it in other problem contexts. For students, Dining Philosophers had memorable imagery and strong material concepts embedded in that imagery that were considered valuable in their problem solving and understanding processes.

"[...] the "dining philosophers" analogy helped explain the concept of deadlock and mutual exclusion by explaining them in terms of dining utensils, ***which are a much more familiar concept.****"* - P7

"The dining philosophers were the most helpful to me. ***I found myself remembering that analogy and referring back to it a lot.*** *This analogy stick in my head because it made the most sense. [...] It was clear and easy to follow. [...] "* - P10

"[...] I liked the dining philosophers analogy just because ***it was kinda funny and quirky, making it easy to remember.****"* - P25

6.1.2 Locking. While not typically considered a classroom analogy, students recognized inherent analogies in syntactic commands when responding to this prompt. Several identified that the concept of "locking" was easy to remember and understand because of its analogous nature to real-world locks. Students referenced lock as an analogy for semaphores. In their programming environment, as well as many other concurrent libraries and environments, "lock" is a literal command or function used to initiate semaphore behavior.

"[...] Thinking of mutexes and semaphores as locks or entrance lines was ***the most broadly applicable analogy*** *for me. [...]"* - P23

"For semaphores, ***the lock and key analogy was super helpful.*** *It helps to think about "left over keys" that others can grab. "* - P28

"The most useful analogies to me are that a semaphore is like a bathroom door lock - ***only the one who locked it can unlock it*** *and it prevents others from getting into the critical section"* - P49

6.1.3 Baton Passing. Another analogy used was that of "baton passing" to represent the concept of processes signalling each other similarly to relay race runners handing off a baton to their next teammate so that teammate can begin their segment of the race. Students indicated the problem was easy to visualize and helpful in explaining an abstract computational process flow. Of note, two responses specifically referenced this analogy as being *easier to understand* than the visual diagrams of the behavior presented in class, which had arrows pointing through the signal relay process.

"I found the baton passing for the semaphores to be helpful in ***learning to trace where the control flow goes*** *because it requires considering multiple semaphores and states at once. [...]"* - P14

"[...] But when they pass the baton it means that they picked up the baton and handed off / signaled another thread. ***This was a better explanation to me than trying to read the slide examples*** *with all the arrows pointing to different parts of the code."* - P50

Baton Passing also elicited this interesting response. One student indicated analogy "falls flat"—but that "naming things effectively" is helpful. Baton passing as a naming convention would still be an analogy to *the act of baton passing*. The student is still referencing the analogy as useful, while describing it as not an analogy:

"Analogies have often fallen flat for me to the point that I don't really remember any good ones from this course. This is really just a personal thing about me. ***Sometimes naming things effectively is what really makes me understand their purpose, like Baton Passing,*** *referring to passing control over a program or critical section to different threads."* - P34

6.1.4 Real World Familiarity. With all of the above themes, students significantly agreed that a large reason for the memorability, and often success, of the analogy was due to a clear, real-world connection. Students indicated that especially with concurrent course material, computing processes can feel quite abstract, and connections to things they could "visualize" aided their understanding.

"I found the analogies for running around a track passing a baton useful for understanding semaphores, because ***it's easy to relate to watching people act something out rather than thinking about how a computer works in technical****"* -P3

"The analogies which describe ***concepts in a more physical sense*** *have been more helpful. [...]"* - P7

"[...] ***Concurrent [...] really has a lot of parallels to real life*** *in the way that threads communicate with each other."* - P18

6.2 Question 4 Response Themes

6.2.1 No Created Analogies. A majority of students indicated that they did not create any new analogies while working to learn the material. While they may have referred to or leveraged the analogies provided in the course materials or problems, the surveys indicated that most did not develop their own analogies.

"I don't think I have developed any of my own analogies for this class. I would refer back to the examples on the slides. Then I would see how I could compare them to the current problem I was working on. ***I have a hard time coming up with my own analogies.****"* - P9

"I did not develop any of my own analogies. The class already had a lot of analogies to study from, which is what I did. ***I think though that coming up with your own analogy would help with understanding the material further.****"* - P24

6.2.2 Course Analogies Useful. While a majority of responses indicated that students did not create their own analogies, several of these responses also suggested this was due in part to feeling the course analogies were sufficient. Students made references to leveraging these analogies and at times even modifying them to suit their specific problem representation needs.

"We did not come up with any new analogies. ***The current analogies that were given were enough*** *to help out in understanding the course material."* - P13

"[...] I was helping a classmate understand the dining philosophers issue since he was having a hard time with it at first. ***We mainly stuck to the analogies provided*** *since they were good examples to being with. [...]"* - P29

"I haven't developed many analogies myself [...] provided in this course were ***sufficient for understanding*** *concepts."* - P38

6.2.3 Competition as Analogy Theme. Among students who did indicate developing their own analogies, there was a variety of unique themes. Among those themes, several responses centered a competitive environment or competition as a significant aspect in the source domain. From sports to conquest tactics, a competition that can be "won" was found across many of the provided analogies.

"Our analogy for ***a Monitor was that it was like a baseball game.*** *There are processes waiting on the bench to bat, and only one process can be at bat at a time. This process can either wait on the next batter by getting a hit and waiting for the next batter/batters to signal him by hitting him home.[...]"* - P27

"[...] compared the problem at hand ***to humans competing or taking turns*** *grabbing things with their hands (the shared resource, or "critical section"). We often thought of the Wait()s and Singal()s as the humans tapping each other on the shoulder and saying "wait, I need to do something" or "I'm done now, go ahead" to each other. This was helpful because when we were acting out these sequences, we would often say these phrases to each other so that we essentially took the role of one of the human threads in the system."* - P46

"[...] For race conditions we thought of it ***like black friday where everyone is rushing to grab items.*** *And sometimes multiple people grab the same item and fight over it which means its a gamble on who gets the item. Eventually one person grabs the item and runs away to buy it so the outcome depends on who gets the item after fighting or who grabs it first. [...] "* - P50

6.3 Question 5 Response Themes

6.3.1 Used in Lower-Level Courses. A significant portion of responses indicated that analogies were leveraged in students' lower-level computing courses. While other courses across the curriculum (and across topic areas, including theoretical computing) were mentioned, responses primarily discussed the introductory programming courses and data structures. Responses concerning these courses will be discussed in the following subsections.

A couple of points are worth noting about the use of analogy in non-introductory courses. Several students mention the use of analogy in lower-level courses when working with others. In addition some students suggest that analogy in lower-level courses helped make novel "abstract" topics more approachable.

"[...] In formal models, I used the analogy of physics ***"matter can be neither created nor destroyed" to help me understand getting rid of lambda rules and chain rules*** *(the chain rule can be replaced with its components so "matter" is simply moved around). "* - P14

"When I first started learning about computer science I found it extremely difficult to understand because a lot of the concepts were so abstract. ***I think it is extremely important to provide analogies when starting to learn material which is typically thought about from a very abstract point-of-view.*** *Most of daily life is talked about using analogies, so it's important that analogies are used when learning new material."* - P47

"I use analogies significantly more in group projects ***when explaining concepts to others.*** *This was very helpful with Team software project and Software Processes and Management [...]"* - P53

6.3.2 Object-oriented Topics. Among introductory programming analogies, students referred to object-oriented topics, such as object design, inheritance, and polymorphism, with the greatest frequency.

"[...] I used ***the blueprint analogy to understand classes and objects****, and used an analogy of exercise (everyone has their favorite activity) to help me understand polymorphism. [...]"* - P14

"I believe I used them a lot in the introductory CS classes. However, I do not remember what they were as it has been a few years. ***It makes sense that they would have been used most in object oriented programming*** *as analogies are the easiest way to envision classes, interfaces, and inheritance.* - P51

Two responses specifically indicate an "Animal" analogy with a petting zoo program for inheritance as particularly helpful:

"[...] I think the one that was ***the most helpful was understanding inheritance using animals****. We would have an animal class which other animals would inherit. This analogy helped me to really quickly understand the concept and the power of inheritance. "* - P8

"There is one particular analogy that I think of from [Intro 2] that really helped me understand object oriented programming. It was a petting zoo program I believe. I remember before that program I was really confused about inheritance, ***but with the animal analogy it made total sense.****"* - P24

These last two responses were particularly gratifying to one of the authors, having developed and presented that assignment.

6.3.3 Data Structures. Another very common theme referred to learning data structures. Students discussed a variety of data structures and analogies that helped them both represent the concepts with something visually concrete, as well as to explain their behaviors. One student even noted how useful analogies are in data structures by citing the analogous names many data structures are given.

"[...] In data structures and discrete, ***I relied a lot on the analogy of a network of roadways to help me understand graphs*** *(the roadways are the edges, there can be multiple edges between nodes/cities, each road has a different cost). [...]"* - P14

"[...] ***We were told to think of a multidimensional array as a shoe holder.*** *There are rows of arrays stacked on top of each other and the empty shoe holder is an array that is initialized but with no data stored in it. The shoes in the holders are the data in the multi-dimensional array."* - P29

"Analogies become particularly useful when talking about data structures. ***Even the names of the data structures themselves*** *imply easy to understand analogies such as a tree, queue, stack, etc. There have been various other analogies [...]"* - P35

"[...] Imagine your grandmother's fine china. If you "stacked" them up, ***you would not want to pull from the bottom and risk breaking all of the china.*** *You would grab one off the top, the last one added. This helped me visualize how stacks actually work without code involved."* - P48

6.3.4 Less Used in Lower-Level Courses. Some responses indicated they felt analogy was used comparatively less in lower-level courses than in concurrent computing. These students tended to suggest that this may be due to concepts being easier, or already understood when presented in their prior courses.

"Analogies haven't been as useful in lower level courses just because ***the content was simpler and easier to understand without them.*** *Learning lower level things like data structures didn't really need analogies to help them be understood. I can't remember any analogies that helped me understand computing concepts in prior*

courses. They were really useful in this one because of the complex ideas that needed simplification. " - P21

"There have been analogies in lower level courses, but not as many as in [Concurrency Course], and none of them really stick out in my mind. ***A lot of the basic concepts (arrays, conditionals, loops, etc) I already understood****, so that might be part of why I don't remember any analogies about them. [...]"* - P37

6.4 Question 6 Response Themes

6.4.1 More Use of Analogy in Upper-Level, More Useful. A majority of responses indicated analogy is *more* useful in upper-level courses, and also that they are using analogies more in these courses.

"I think they are more useful as an upper-level student. ***Lower level classes have simpler ideas, and often times issues are in the technical detail, not the big ideas.*** *They are much more effective in [third or fourth year] level classes in my view. "* - P5

"Oh I definitely use them more. A whole lot more. I have to. ***I mean, I don't see another way of doing things. I really don't. For me, analogies are all but mandatory for understanding code.*** *If I run across a piece of code that I cannot break down into some kind of analogy then I'm probably not going to be able to understand it. I need to be able to physically see something happening in front of me (in my head) in order to properly code it myself[...]"* - P9

6.4.2 Less Use or Same Amount in Upper-Level. Some responses indicated an appreciation for the value of analogy across course levels. Some even suggest that while they no longer use outside domain sources for analogs, they may use *computing concepts as source domains* to make sense of material.

""I feel like analogies are effective for both upper and lower level students. They're useful for lower level students because they're just learning the basics and don't have much of a background in computing. Analogies help tie the concepts they're learning to ideas they can better understand. ***Analogies can be useful for upper-level students, but I don't think it's as helpful as it is for lower-level students.*** *Upper-level students have an understanding of computing, so it's easier for them to understand new concepts without analogies."* - P31

"[...] however as I have grown as a CS student I have found that computing concepts in themselves are very comfortable to me and in fact ***now when trying to learn other skills I use CS analogies to help me.*** *the early years were harder because I was less familiar with how computers worked so I needed more tools to help me"* - P19

6.4.3 Useful in Explaining, but Take Care in Usage. Students also reflected on the value of analogy as an explanatory tool, both in the positive, and in the cautionary. They indicated a sense of analogy being a useful tool in explaining concepts, but indicated that caution should be taken to make sure that appropriate analogies are being developed and leveraged when explaining.

"I feel like I understand the importance of using analogies when trying to explain something to someone. ***I would say I use them much more now when explaining things to people than I did in the past.*** *[...]"* - P47

"[...] ***analogies are very helpful if they are relevant and well thought out*** *otherwise they can be tedious and confusing."* - P51

6.4.4 Difficult to Create as a Student. Several students also indicated difficulty in creating their *own* analogies, but suggested that being able to might showcase or improve their understanding.

"[...] ***If I was better able to come up with my own analogies, I think it would help to better understand the material for a class.*** *If you can make an analogy for a problem, then you must understand the material enough to do so [...]"* - P10

"I feel that when I manage to make them, they are very helpful. ***It is somewhat difficult to make a good analogy for something*** *because you have to be able to cover all of the complexity of the problem you are talking about while also still being able to relate it to what you are working on. [...]"* - P51

7 DISCUSSION

Across these four questions and their responses, students explored several topics worth consideration across the computing curricula.

Analogies Get Used Across Computing Curricular Topics. Students readily recognized analogies from courses they had taken years prior when answering Question 5, and in Question 6 reflected on the effectiveness of analogies they had seen across their time learning computing topics. Upper-level students, who are able to "look back" at their time in our programs, are in a wonderful position to tell us what is happening within their courses. While we recognized some analogies which may appear in prior courses, we were surprised to find the number of distinct courses, contexts, and analogies students recalled. Students also recognized that *many types of pedagogical activities are analogies*, such as representation through a modelled/visual simulation, programming assignment, or hands-on physical activity.

Students Recognize Analogies Embedded in Our Terminology and Examples. We were intrigued to see how many students specifically indicated core terminology and seminal examples as analogies they recognized. A "tree" data structure or "lock" function are deeply ingrained analogies within the discipline, yet students are still able to recognize these are analogous, rather than strictly novel, naming conventions. This is in clear contrast to arguments in the same vein as Dijkstra [11] which argue the "radical novelty" of our field is antithetical to analogy. Within modern computing, our very terminology often rests upon it. Further, seminal examples within our field, such as "Dining Philosophers" (ironically, developed by the very same Dijkstra [10]) or "Travelling Salesman", are deeply analogous by design and leveraged across institutions.

Students See Creating Analogy as a Challenging Activity Indicating Mastery. Across questions, particularly 4 and 6, students indicated a sense that analogy creation was a difficult act. These students often suggested their personal sense of struggle to develop analogy. Responses indicating this difficulty also often acknowledged in parallel the belief that *ability to create analogy corresponded to understanding material.* This suggests that students find creating analogies to be an "expert" task. However, taken from an instructional perspective, we might suggest that this belief begs the question: *If student analogy development or exploration was part of a pedagogical activity, would they feel more confident in their understanding or increase that understanding through the process?*

Analogies are Useful for Explanations, but How Often are Students Explaining? Students also showcased seeing analogy as a useful communication and explanation tool. Even if they struggled to create analogies, many indicated using known analogies to explain concepts to friends. However, students did not appear to often place themselves in situations where they were likely to need to communicate with others. Several responses indicated working solo, often working to understand material by oneself, or not being a part of any study groups (this theme was omitted from the results section in the interest of space, but worth noting for discussion here). This was somewhat surprising to us, as our concurrency course is considered one of the most difficult courses in the major—we expected to some degree that students might more readily collaborate for mutual success. Taken together with the above theme, we are also left wondering if some of the difficulty in analogy creation might come from not being in *spaces where analogies are likely to be generated.* Given the belief that analogy creation can be indicative of subject understanding, we wonder: *If students were overall more involved in collaborative environments, would they be more likely to develop new analogies alongside existing ones?* This question arises from responses by students who indicated they *were* in study groups. Many of them shared ways they developed new analogies, or how they scaffolded new ideas onto existing analogies in order to help explain and understand aspects that were confusing to some group members.

Analogies can be Really Impactful and Memorable to Students. The novelty of analogies and sometimes "quirky" source domain choices can be looked down upon by some as flash-in-the-pan shock value, fading away with little substance. However, many students recalled in vivid detail how specific analogies catalyzed deeper understanding of the a topic. While not every student may have this experience, we see that for many, an analogy can serve as the "Eureka" moment in unlocking understanding. This is not to say that these students rely on the analogy exclusively—rather, the analogy created a pathway, or scaffold, to build deeper understanding of the concept. This suggests evidence of analogy aiding in long-term transfer, which has not appeared to be previously identified [5]. In this same way, we might wonder: *What other analogies were not remembered, but were a significant part of the scaffolding process, eventually fading away to conceptual understanding?*

Analogy Isn't Everyone's "Favorite" Approach, and That's Okay. We would be remiss if we were to suggest that all students feel fondly about analogies. While we were surprised to find many stories of impact and value, students are not all the same, and not every student felt analogies were useful for them. There was often disagreement: while many students found "baton passing" to be very useful, at least one student indicated it was a terrible analogy in a space that desperately needed a good one. Not every analogy works for every student, and not every student finds analogy to be the most impactful approach. We were somewhat surprised to find that some students who found analogy less impactful directly indicated analogies as impactful in other ways, such as the indication that analogies were not useful but that baton passing was a "fitting naming convention" and thus useful. This might suggest that some students have preconceived notions of what an analogy is, or the utility of them. Regardless, there will always be some students who do not find analogy to be as valuable of a pedagogical practice for them, and we saw evidence of this. Several of these students also indicated that while analogy might not be "for them", they could see value for others. Our students sum it up best in this way: leveraging multiple pedagogical practices can help ensure many pathways of understanding for different types of students. Analogies don't have to be everyone's favorite to be valuable, and analogy's value does not have to be an exclusive-or in relation to other approaches. The more ways our students have to explore novel material, the better chance each student will find a path to understanding.

8 CONSIDERATIONS AND LIMITATIONS

Our survey was deployed at one university during a single semester of the concurrent computing course. Demographic data was not collected. Collection of demographic data may have allowed us to further analyze response trends, since demographics may impact how a student responds. The course was also conducted online for the second half of the semester due to the COVID-19 pandemic, and this may also impact responses. We also recognize that our materials and project descriptions do make use of analogy, and that different pedagogical approaches may provide distinctly different responses.

9 FUTURE WORK

We would like to gather further data from new cohorts of the concurrent computing course. Such data could be independently thematically analyzed, as well as compared to prior analysis (such as this work). Future forms of the survey may benefit from demographic questions, which would also allow new analysis and new research questions. We hope to further explore questions that this work has revealed, such as those posed in our Discussion section, through future studies.

10 CONCLUSION

This paper examined perspectives of students in an upper-level concurrent computing course on analogy usage within the course and in their previous computing courses. Thematic analysis was conducted on four qualitative survey questions centering ideas of analogy use across the course and in prior courses. Student responses showcased recognition of analogy use across the computing curricula, including the often "taken for granted" analogies embedded in our field in terminology such as "tree' and "lock". Students indicated a sense that analogy is a powerful explanatory tool, but one that they see as challenging to develop themselves. They were able to recall specific examples in detail that aided their understanding of computing concepts from across their years, showing potential evidence of long-term transfer and value in understanding computing concepts. While analogies may not be the perfect method for every student, they are certainly a method which many students identify as helpful. This exploration suggests that continued work in exploring analogy *across* computing curricula would be beneficial, and that further research specifically involving upper-level students and analogy—a quite under-explored area within CER—be conducted.

REFERENCES

[1] Thomas Anderson and Michael Dahlin. 2014. *Operating Systems: Principles and Practice* (2nd ed.). Vol. Volume II: Concurrency. Recursive books.

[2] Briana Bettin. 2020. The Stained Glass of Knowledge: On Understanding Novice Mental Models of Computing. , 307 pages. https://digitalcommons.mtu.edu/etdr/1086/ Open Access Dissertation.

[3] Briana Bettin and Linda Ott. 2021. Frozen in the Past: When It Comes to Analogy Fears, It's Time For Us to "Let It Go". In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE '21)*. Association for Computing Machinery, New York, NY, USA, 359–365. https://doi.org/10.1145/3430665.3456381

[4] Briana Bettin, Linda Ott, and Julia Hiebel. 2022. 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)*. Association for Computing Machinery, New York, NY, USA, 200–206. https://doi.org/10.1145/3502718.3524796

[5] Yingjun Cao, Leo Porter, and Daniel Zingaro. 2016. Examining the Value of Analogies in Introductory Computing. In *Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER '16)*. Association for Computing Machinery, New York, NY, USA, 231–239. https://doi.org/10.1145/2960310.2960313

[6] Kathy Charmaz. 2014. *Constructing Grounded Theory*. SAGE Publications.

[7] Yam San Chee. 1993. Applying Gentner's Theory of Analogy to the Teaching of Computer Programming. *Int. J. Man-Mach. Stud.* 38, 3 (March 1993), 347–368. https://doi.org/10.1006/imms.1993.1016

[8] Thiago de Jesus Oliveira Duraes, Paulo Sergio Lopes de Souza, Guilherme Martins, Davi Jose Conte, Naylor Garcia Bachiega, and Sarita Mazzini Bruschi. 2020. Research on Parallel Computing Teaching: state of the art and future directions. In *2020 IEEE Frontiers in Education Conference (FIE) (FIE '21)*. 1–9. https://doi.org/10.1109/FIE44824.2020.9273914

[9] Barbara Di Eugenio, Nick Green, Omar AlZoubi, Mehrdad Alizadeh, Rachel Harsley, and Davide Fossati. 2015. Worked-out Examples in a Computer Science Intelligent Tutoring System. In *Proceedings of the 16th Annual Conference on Information Technology Education (SIGITE '15)*. Association for Computing Machinery, New York, NY, USA, 121. https://doi.org/10.1145/2808006.2808011

[10] Edsger W. Dijkstra. 1971. Hierarchical ordering of sequential processes. *Acta Informatica* 1. https://doi.org/10.1007/BF00289519

[11] Edsger W. Dijkstra. 1988. On the cruelty of really teaching computing science. (Dec. 1988). http://www.cs.utexas.edu/users/EWD/ewd10xx/EWD1036.PDF Circulated privately.

[12] Michal Forišek and Monika Steinová. 2012. Metaphors and Analogies for Teaching Algorithms. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE '12)*. ACM, New York, NY, USA, 15–20. https://doi.org/10.1145/2157136.2157147

[13] Klaus-Tycho Förster, Michael König, and Roger Wattenhofer. 2016. A Concept for an Introduction to Parallelization in Java: Multithreading with Programmable Robots in Minecraft. In *Proceedings of the 17th Annual Conference on Information Technology Education (SIGITE '16)*. 169. https://doi.org/10.1145/2978192.2978243

[14] Dedre Gentner. 1983. Structure-mapping: A theoretical framework for analogy. *Cognitive Science* 7, 2 (1983), 155 – 170. https://doi.org/10.1016/S0364-0213(83)80009-3

[15] Nasser Giacaman. 2012. Teaching by Example: Using Analogies and Live Coding Demonstrations to Teach Parallel Computing Concepts to Undergraduate Students. In *Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW '12)*. 1295–1298. https://doi.org/10.1109/IPDPSW.2012.158

[16] Mark Guzdial. 2020. Dijkstra Was Wrong About 'Radical Novelty': Metaphors in CS Education. https://cacm.acm.org/blogs/blog-cacm/248985-dijkstra-was-wrong-about-radical-novelty-metaphors-in-cs-education/fulltext

[17] Jesper Haglund. 2013. Collaborative and self-generated analogies in science education. *Studies in Science Education* 49 (03 2013), 1–34. https://doi.org/10.1080/03057267.2013.801119

[18] Rachel Harsley, Nick Green, Mehrdad Alizadeh, Sabita Acharya, Davide Fossati, Barbara Di Eugenio, and Omar AlZoubi. 2016. Incorporating Analogies and Worked Out Examples As Pedagogical Strategies in a Computer Science Tutoring System. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 675–680. https://doi.org/10.1145/2839509.2844637

[19] David James. 2020. The Use of DJing Tasks as a Pedagogical Bridge to Learning Data Structures. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20)*. Association for Computing Machinery, New York, NY, USA, 193–197. https://doi.org/10.1145/3341525.3387427

[20] Elizabeth R. Koning, Joel C. Adams, and Christiaan D. Hazlett. 2019. Visualizing Classic Synchronization Problems. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (2019). https://doi.org/10.1145/3287324.3293708

[21] Joseph P. Sanford, Aaron Tietz, Saad Farooq, Samuel Guyer, and R. Benjamin Shapiro. 2014. Metaphors We Teach by. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. ACM, New York, NY, USA, 585–590. https://doi.org/10.1145/2538862.2538945

[22] Ching-Kuang Shene and Steve Carr. 1998. The Design of a Multithreaded Programming Course and its Accompanying Software Tools. *Journal of Computing in Small Colleges* 14, 1 (1998), 12–24.

[23] Klaas-Jan Stol, Paul Ralph, and Brian Fitzgerald. 2016. Grounded Theory in Software Engineering Research: A Critical Review and Guidelines. https://doi.org/10.1145/2884781.2884833