Spring 3-19-2020

Communicating Computing Limitations Through Kinesthetic Pedagogy

Michael Mason
University of Nebraska - Lincoln

Communicating Computing Limitations Through Kinesthetic Pedagogy

An Undergraduate Honors Thesis
Submitted in Partial fulfillment of
University Honors Program Requirements
University of Nebraska-Lincoln

by
Michael Mason, BS
Computer Science and Mathematics
College of Arts and Sciences

March 19, 2020

Faculty Mentor:
Charles Riedesel, PhD, Computer Science

Abstract

Abstract concepts, such as those in advanced Computer Science and Mathematics, can be extremely difficult to understand fundamentally without an existing background in a similar subject (Hartenstein 2010). Recent research has shown that raw visualizations without learner interaction are not particularly effective at communicating complex information because they allow the learner to ignore the example (Lauer 2006, Naps 2002). Forcing somebody to interact with an example ensures that they can grasp the visualization (Sivilotti and Pike 2007). This paper describes a six step technique to demonstrate the limitations of computing through kinesthetic pedagogy, then offers an example exercise utilizing the method. The six proposed steps are: define the problem, identify the core computation needed, model this problem and its core computation with something that is easy to interact with, have the audience interact with the model, elicit a concession of difficulty, and finally return model to the problem. This process clearly connects abstract concepts to a physical interpretation, and provides an intuitive avenue for learning. Although further testing is required to determine the process's effectiveness, previous research has shown that similar pedagogical techniques have dramatically increased learning (Sivilotti and Pike 2007, Goldsmith and Mihail 2016).

Key Words: Engineering Education, Computer Science Education, Cross-Disciplinary, Science Communication, Computing Limitations, kinesthetic pedagogy, parallel processing.

Introduction

Computer systems are continuing to increase in complexity, implement more abstract mathematical concepts, and are rapidly expanding into every part of life (Hartenstein 2010). As the range of roles computers are required to serve expands, the number of people without formal computer training that interact with complex computer systems continues to grow (Hartenstein 2010). This becomes particularly troublesome when new systems are proposed by individuals that do not understand how their plans must be implemented, and causes problems when developers are asked to create solutions which are impossible or unrealistic (Brooks 2013). An example of this situation is the Clinton administration's Clipper chip, a component that would allow law enforcement and only law enforcement to wire tap any device that had a Clipper chip installed. This is reflected in the more modern concept of a backdoor in smartphones that only "good guys" can access (Rodriguez, 2019). These security protocols are impossible to perfectly implement, but it took a considerable amount of time for policy makers to come to this conclusion in part because of ineffective communication techniques (Bacard 1995).

To address the problem of unrealistic development proposals, techniques must be produced that allow educators and technical specialists to give an intuitive explanation of computing limitations to individuals without the strong background in computing typically required to understand a task's feasibility. This paper describes a six step method aimed at making it clear to an audience why specific problems are not feasible computationally without requiring them to understand the rationale on a deeper level. The proposed method utilizes kinesthetic pedagogy over the less effective practice of providing a visualization because it forces the audience to interact with the example, whereas raw visualizations can be ignored (Sivilotti and Pike 2007). A direct application to programming education is outside the scope of this paper because we are specifically interested in a communication technique to present high level computing concepts that does not require any prior programming or computer expertise, although it may still be applied to programming concepts. The paper begins by describing the six steps, then justifies choices made in each step, and finally

provides an example implementation of the process to model a parallel computing limitation.

The Six Steps Described

This section details the six steps to the proposed method in detail. The six steps are: define the problem, identify the core computation needed, model this problem and its core computation with something that is easy to interact with, have the audience interact with the model, elicit a concession of difficulty, and finally return the model to the problem. In addition to these six steps, there are common cross-disciplinary communication techniques that should be utilized at all points. The most important aspect to consider is audience analysis. The specific model may need to change based on audience, but this six step framework remains viable regardless of who the presentation is being performed for. Avoid disciple-specific terms outside of common vocabulary, also known as jargon (Smith-Worthington and Jefferson 2018). If jargon must be used, it should be clearly defined the first time it is utilized. Consider the education level of the audience with particular attention paid to their previously received training (Smith-Worthington and Jefferson 2018). Finally, consider assumptions made throughout the process (Smith-Worthington and Jefferson 2018). Each step will now be described in detail.

The first step is to clearly define the problem. Identify all parameters including the overarching goal of the problem, average input, edge-case input, cost constraints, time constraints, space constraints, expected output, possible output, and any other relevant information. The identified problem will either be feasible, infeasible, or impossible to solve. If the problem is feasible, then it is outside the scope of the proposed method and is not addressed again. Determine if the problem is infeasible or impossible, then clearly state to the audience which of the two options it is. Progress to the second step.

The second step is to identify the computation needed to solve the problem. This begins by determining the steps required to solve the problem. Explain an algorithm in simple terms which will solve the desired problem as efficiently computationally as possible. Note why this is the most

efficient algorithm, likely identifying the algorithm as a previously researched solution which has a known time complexity. After identifying an algorithm, you must determine the core computation required to perform the algorithm which makes it infeasible or impossible to implement. Give a direct statement such as: “What I’m about to explain to you is the underpinning of why this cannot be done. These problems are the same because...”. Progress to the third step.

The third step is to reduce the problem to something which is easily modeled kinesthetically. The model created in this step does not directly tie to the identified problem, but rather models the core computation required to solve the identified problem. Begin by introducing the model exercise by identifying the physical pieces involved, the steps that these pieces must go through, and the goal of the exercise. After defining the exercise to the audience, directly connect each physical piece, process step, and goal to the core computation and algorithm defined in step two. Additionally, directly state after connecting each piece of the model exercise to the parameters from step two that the identified problem from step one is solved by the core computation in step two that is being modeled now, so this model exercise directly represents the identified problem. Progress to the fourth step.

The fourth step is to have the audience directly interact with the model. The audience’s interaction with the model must not contain any shortcuts; they must explicitly perform each step that has been ported over from the algorithm, and the person directing the exercise may not perform any part of the simulation. If data is required to elicit certain conclusions in the fifth step, that data must be gathered during the audience’s interaction with the model in this step. Progress to the fifth step.

The fifth step is to elicit a concession of difficulty from the audience. This is done by asking leading questions based off of their experience with the exercise and the data gathered in step four with the goal of having the audience coming to the conclusion that the model exercise in some way

has a problem which does not allow the desired solution in the identified problem from step one to exist. These questions will vary depending on the model exercise, but there are three questions that should be asked no matter what the model is. The first question or questions must address the exercise and their experience with it. Have the audience identify the limit that appeared within the exercise that makes the problem identified in step one infeasible or impossible. Do not directly ask them about the problem itself, but rather frame the question with respect to the exercise. The second question to pose is how the operations in the exercise could have been altered or performed in a different way that would cause the infeasible or impossible problem to be solved more easily. This step is completed by asking the third question: ”Do you understand why this task is infeasible/impossible?” If their answer is yes, then ask the audience to explain why they believe it is and proceed to the final step if the answer shows sufficient understanding. If they answer no or they do not show sufficient understanding after answering yes, then either return to the beginning of step four if their experience did not accurately reflect the algorithm or improper data collection occurred, or return to the beginning of step five if the audience did not draw sufficient conclusions based on the post-exercise discussion. Statements of fact or conclusions the audience should have drawn provided by the educator should either come after they have drawn the conclusion themselves, or after the posed questions do not receive an answer. In the case that posed questions do not receive an answer, it is more advantageous to reformulate or further subdivide the questions until responses are obtained. Progress to the sixth step.

The final step is to return the model to the problem. This step can be summarized as working backwards from step three to step one. Reconnect each piece from the model exercise to the core computation and algorithm used, and remind the audience that they concluded that the exercise had an infeasible component, which makes the core computation from step two have an infeasible component. Finally, reconnect each piece from the core computation the the originally identified problem, reminding the audience that there was an infeasible component from the core computation, so there must be an infeasible component in the identified problem. After explicitly stating

that this modeling process implies that the identified problem has an infeasible component, the process is completed.

Justification of the Six Steps

This section provides justification as to why the six steps are necessary to solve the issue of implementing kinesthetic pedagogy to teach computing limitations. Each step is justified individually.

The first step is to clearly define the problem. This step is necessary in order to ensure that all participants involved understand what the motivating problem is, and verifies that all audience members have the same problem and parameters in mind when participating in the exercise (Smith-Worthington and Jefferson 2018). Spending time to identify the problem not only adds structure to the process, but also helps prevent confusion later on because all parties involved know what the goal and parameters are (Smith-Worthington and Jefferson 2018). Clearly defining the feasibility of the problem acts as a thesis statement for the whole process after providing background by identifying the problem. Creating a strong statement to use as a thesis motivates the audience to listen because it creates a strong connection between the goal of understanding the exercise, and creates a direct tie between the exercise and the problem at hand (”Engagement Practices” 2020).

The second step is to identify the computation needed to solve the problem. This step is utilized to simplify the problem in a way that is more easily represented with a model. Identifying the core computation and using it to represent the overarching problem strips away irrelevant details that do not affect weather or not there exists a feasible solution. Additionally, simplifying the problem to a core computation allows the use of previous research that has been done in Computer Science. By using previous research in Computer Science, it is possible to deliver known solutions, however feasible, that are backed by research institutions, and are therefore assumed to be correct and efficient. Citing where the algorithm was obtained further enforces this idea in the mind of the audience. Finally, providing a direct statement which explains that the identified problem is

at least as hard as the core computation described shows the audience that what they are about to experience is relevant to the process.

The third step is to reduce the problem to something which is easily modeled kinesthetically. This step is unique from the majority of other pedagogical methods because it does not allow for a traditional visualization, but rather forces audience interaction so they are unable to ignore the explanation. The model created in this step should not be directly derived from the original problem to ensure that details of the identified problem do not distract from the core computation's relationship with the model exercise. By directly tying each part of the model in this step to a parameter in step two, the audience makes a logical connection between the problem and the exercise. Stating this explicitly not only prevents audience members from not connecting the steps together logically, but reinforces that the identified problem is equivalent to what they are about to perform.

The fourth step is to have the audience directly interact with the model. Again, this step is what makes this model unique from the majority of other pedagogical approaches. As noted previously, raw visualization of an algorithm or data does not guarantee audience attention (Lauer 2006, Naps 2002). Audience members must interact with the exercise themselves to ensure that this is truly a kinestetic experience rather than a visual one. Data gathered during this stage is produced by the audience's own hand, which gives an intuitive rationale as to why the data exists as it is.

The fifth step is to elicit a concession of difficulty from the audience. By asking questions, this step employs the Socratic method that has been used for centuries as an educational tool to extract contradictions that lead to realizations (Garrett 1998). The first question or questions must address the exercise and their experience with it. This question is required in order to extract information and conclusions that the audience has drawn through the course of the exercise. Without asking them about the model exercise, there is little point in having the exercise executed at all. Additionally,

asking them questions about their experience with the model aids in transitioning from the exercise to this step. Asking how the algorithm could be more efficient makes the audience think about the exercise in detail, and if they come to the conclusion that there is nothing they could do to make it better it will aid them in the conclusion that this limit truly exists. The third and final question assures that the exercise proved its point, and the corollaries following that question ensure that if the exercise did not prove its point, that it is reached eventually.

The final step is to return the model to the problem. This is necessary to strongly connect the concession of infeasibility from the fifth step to the original problem. Working backwards and strongly connecting each step unifies the exercise and the supporting background information introduced through the process. Explicitly stating that the model implies that the identified problem has an infeasible component that they have personally identified as infeasible removes all room for ambiguity.

Example Application of the Method

This section details an example application of the described method. The limitation to be animated is the computing wall provided by the limit of Amdahl's law that prevents more rapid computation by adding additional cores past a certain point because of the overhead required to distribute and recombine the problem which was parallelized (Silberschatz et al. 2012). This is a relevant problem to be addressed because modern computation is becoming more focused on cloud and multicore computing (Silberschatz et al. 2012).

Before the the six steps, consider audience analysis and assumptions made. This exercise requires a substantial number of people, preferably 20 to 30 to simulate numerous cores, and is therefore not effective at demonstrating Amdahl's limit to small groups. It is particularly relevant to business stakeholders that are trying to find ways to improve processing speeds of their computer infrastructure, and students learning concepts related to Amdahl's limit. This exercise assumes thread safety and no synchronization issues like race conditions, which would allow participants to accidentally

interfere with another participant's model thread. It also assumes that all problems can be threaded. This exercise is digestible for an audience of any skill level.

The first step is to clearly define the problem. Begin by explaining what a core is, and that threads are sub-portions of a process that are individually allocated to each core for execution. The problem to be demonstrated is that adding additional cores or threads has a diminished rate of return with respect to computation speed of a problem that has been split for such processing, and that computation is not infinitely scaleable with a larger, more powerful computer. State that this is a relevant problem because most modern computers are multithreaded, and clearly connect it to the motivation for utilizing this exercise. Note that this is assuming that a single line of processing is less quick than multiple threads processing information at the same time because otherwise there is no point to splitting the computation. Parallelization is a design scheme rather than a specific problem to be solved, so there must be a token problem that is solved in each thread that can easily be split and recombined into one overarching problem. Because of this, you must state to the audience that a sample problem is being used to animate the true problem in this circumstance. One such problem that can be parallelized is sorting. Note to the audience that the new problem being solved is sorting with multiple threads. Tell the audience that it is infeasible to drastically increase computation speed using hardware past a certain point because of this problem, and it is impossible to push computation between a particular limit at all. Progress to the second step.

The second step is to identify the core computation needed to solve the problem. The core computation needed for sorting is a comparison between two objects, but the more important operation that must be performed is splitting the input between and recombining the product of each core together after their individual computation. The specific sorting algorithm utilized by participants is not relevant, and will be developed by the audience organically during the exercise. Note to the audience that sorting is relatively quick and intuitive for humans, and the most difficult part will be coordinating all of their efforts together. Explain to the audience that the algorithm to be

implemented contains two steps per iteration: sort the portion of the overall list allocated to your thread, then recombine your thread's information with other threads that have completed sorting. Progress to the third step.

The third step is to reduce the problem to something which is easily modeled kinesthetically. A standard deck of playing cards is a good physical representation of a list to be sorted because each card is marked with symbols that indicate a clear order, each card can be individually manipulated, and for our purposes the cards can be distributed among multiple people. For this exercise, the cards are considered sorted when they are in order from ace to king and grouped by suits in order of spades, clubs, diamonds, and hearts. The number of iterations of the exercise depends on the number of participants. Beginning with a single person, have them sort the whole deck after it is shuffled. At each iteration divide the number of cards each person has by two, and give an additional person the other half of their cards. Stop after each person is involved in the exercise. If there are not enough cards to distribute a card to every person, begin having two or more people sort a single card. A person cannot begin to recombine their cards with another person until both participants have their individual set of cards completely sorted. This gradually models sorting the deck of cards with a single core to many cores. Provide a direct statement such as: "What I'm about to have you do is the underpinning of why computing power cannot be infinitely increased by adding more processors. These problems are the same because you will be simulating a number of cores, and will reach the conclusion that at a certain point adding additional people does not meaningfully increase the speed at which you sort." Reiterate to the audience that sorting the cards represents a process, and each person represents a core. Progress to the fourth step.

The fourth step is to have the audience directly interact with the model. Have the audience perform the exercise as described in step three. At each iteration, make note of the number of people sorting and the amount of time it takes from start to finish.

The fifth step is to elicit a concession of difficulty from the audience. Begin by asking questions such as how recombining the smaller lists was performed and what challenges it created, what happens if there are more people than cards to be sorted, who had priority in how the lists were recombined, and what happened when a participant finished before another sorter and how that impacted the amount of wasted time. Plot the data gathered in the fourth step on a Cartesian plane with one axis representing number of sorters and one axis representing the amount of time spent on that iteration of the exercise. The graph will be a curve that quickly approaches an asymptote. Ask an audience member to draw a line on the graph which they think that the computation can can never go faster than regardless of the number of cores. Next, ask the audience for suggestions that would make the implemented algorithm run faster. Furthermore, if they make a suggestion for improving the algorithm show how it does not alter the base problem of distributing and recombining tasks. A variant of this exercise involves having a single person direct who is allowed to combine with who as an individual finishes sorting their list. If this is not utilized during the exercise, pose as a question what would happen if there was such a person. Adding a director does not speed up individual computation, and would only minimally improve recombination of the list. Finally, ask the audience if they understand why this task is infeasible or impossible to improve by adding more cores. When they respond favorably by identifying that the greatest overhead may not lie in computation, but in thread distribution and recombination which prevents a computer from infinitely scaling its speed with additional threads, move to the final step.

The final step is to return the model to the problem. Explain to the audience in reverse order from the exercise they performed with the cards demonstrate how parallel computing works. Note that the cards and the individuals sorting them directly represent a core sorting a list of elements, that a cores and a list of elements to be sorted in parallel is a parallel computing problem which implements multiple cores, and finally note to the audience that this is an instance of a parallel computation which has a maximum speed limit imposed on it by Amdahl's law because adding additional cores has a diminishing rate of return for computation. This concludes the final step in

this process to model Amdahl's law through kinesthetic pedagogy.

Conclusion

This paper described a six step method to implement a kinestetic pedagogical approach to demonstrating limitations imposed on computing. The paper begins by describing the six steps, then justifies choices made in each step, and finally provides an example implementation of the process to model a parallel computing limitation. Additional research is required to confirm the validity of this approach, but similar methods utilized for teaching data structures and programming techniques have dramatically increased learning (Sivilotti and Pike 2007, Goldsmith and Mihail 2016).

Refrences

Anon. Engagement Practices. Retrieved March 20, 2020 from https://www.engage-csedu.org/engagement/make-it-matter

Andre Bacard. 1995. *The Computer Privacy Handbook*, Berkeley, CA: Peachpit Press.

Frederick Phillips Brooks. 2013. *The Mythical Man-month: Essays on software engineering*, Boston, MA: Addison-Wesley.

Elizabeth Garrett. 1998. The Socratic Method. (1998). Retrieved March 20, 2020 from https://www.law.uchicago.edu/socratic-method

Judy Goldsmith and Radu Mihail. 2016. Kinesthetic Touches For a Theory of Computing Class. *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS)* (2016), 199–204.

Reiner Hartenstein. 2010. Reconfigurable Computing: boosting software education for the multi-core era: Why we need to reinvent computing. *2010 VI Southern Programmable Logic Conference (SPL) (2010)*. DOI:http://dx.doi.org/10.1109/spl.2010.5482991

Tobias Lauer. 2006. Learner Interaction with Algorithm Visualizations. *ACM SIGCSE Bulletin 38, 3* (2006), 202. DOI:http://dx.doi.org/10.1145/1140123.1140179

Thomas L. Naps et al. 2002. Exploring the Role of Visualization and Engagement in Computer Science Education. *Working group reports from ITiCSE on Innovation and technology in computer science education* - ITiCSE-WGR 02 (2002). DOI:http://dx.doi.org/10.1145/960568.782998

Fidel Rodriguez. 2019. Ann Arbor, *Hidden Backdoors.* MI: ProQuest LLC.

Abraham Silberschatz, Peter B. Galvin, and Greg Gagne. 2012. *Operating System Concepts Essentials*, Hoboken, NJ: Wiley.Paolo A.G. Sivilotti and Scott M. Pike. 2007.

Darlene Smith-Worthington and Sue Jefferson. 2018. *Technical Writing for Success*, Boston: Cengage Learning.

Paulo Sivilotti, Scott Pike. 2007. The Suitability of Kinesthetic Learning Activities for Teaching Distributed Algorithms. *Proceedings of the 38th SIGCSE technical symposium on Computer science education - SIGCSE 07* (2007). DOI:http://dx.doi.org/10.1145/1227310.1227438