Spectrum of Engineering Sciences
Vol-1, Issue-2
http://thecjlr.online/index.php/1/issue/view/2

Examining the Importance of Visualization and Engagement in Computer Science Education

Asim Usman
Ms Scholar Department of Computer Science the Sarhad University Peshawar
Muhammad Babar
Ms Scholar Department of Computer Science the Sarhad National University Peshawar

Abstract

The application of visualization technology allows for the visual representation of a wide variety of computer science concepts. No matter how effectively it is constructed, we argue that such technology has very little educational value unless it actively engages students in learning. This is experimental investigations that investigated the efficacy of visualization lends support to this stance. Regarding the utilization of visualization, it is located within the framework of the prevalent attitudes and best practices that are now in place. The authors offer a new taxonomy for the classification of learner engagement with visualization technologies. We provide metrics for evaluating the educational successes that may result from such active participation, and we do so by building upon Bloom's taxonomy of comprehension, which is widely recognized. Using these taxonomies of effectiveness and engagement measures as a foundation, we present a framework for conducting experimental investigations into the effectiveness of visualization. Educators in the field of computer science who are interested in working together with us to do research within this framework are cordially invited to do so.

Keywords- Visualization and Engagement in Computer Science, Education

Introduction

This paper is the outcome of the Working Group on Improving the Educational Impact of Algorithm Visualization's cooperative efforts. Under the direction of Tom Naps and Guido Roßling, the group's operations were started in the spring of 2002. Through the use of a groupware application and a listserv mailing list, the group was able to generate a draft report, administer an online survey, and hold discussions on various topics. These tasks were finished and completed prior to the group's in-person assembly at the ITiCSE conference in Örhus, Denmark. The terms "we" and "they" will hereafter be used to refer to the Working Group, which is made up of the people who were stated in the opening of this report. Furthermore, there is a section in this report that thanks the three remote members that attended. The introduction of visuals into the realm of computation is motivated by the abstract nature of the fundamental components of computation. Based purely on intuition, graphical depictions of these architectural elements might simplify understanding of their operation by presenting them in a more tactile format. The main goal of visualization software when it was originally released in the late 1980s was to provide graphical representations of computer science subjects that could be investigated interactively [8, 54]. Recent surveys of computer science educators have revealed that there is broad consensus regarding the beneficial effects of visualization technology on learning. However, the experimental studies that were conducted to determine the educational efficacy of such visualization technologies do not corroborate this claim [28].

The results of our pre-conference poll led us to the conclusion that another significant barrier to the adoption of visualization is the massive time and effort that educators must devote to integrating technology into their instruction. The two primary barriers that have been found to

be impeding the mainstream application of visualization technologies are as follows: The student may have doubts about the visualization technology's ability to teach effectively. From an educational perspective, there could be significant overhead related to the visualization technology, making it unprofitable. What efforts may be taken to address these problems, given that computer science educators are confident that, when used successfully, visualization technologies may be of significant use to both teachers and students? It makes perfect sense to focus more on the results of earlier experimental studies when examining the efficacy of education.

When using the visualization technology actively, learners often outperform those who merely view the visualizations in terms of scores [28]. After a closer look at those research, a significant trend becomes apparent. The following activities, for the sake of illustration, have been effectively completed by students using visualization technology: creating their own input data sets (40, Chapter 9); making predictions about future visualization states (11); programming the target algorithm (31); responding to strategic questions about the visualization (23, 44); and creating their own visualizations (26). Given this pattern, it seems sense to design further experimental studies that examine the deeper and more in-depth educational impacts of different kinds of active involvement. We present an outline of a framework for carrying out this type of empirical research in the report that follows. Such research can be carried out in classroom settings as well as controlled facilities due to the framework's adaptability. We contend that until visualization technology actively engages students in learning, its instructional effectiveness is limited, regardless of how effectively it is built. Regardless of how beautifully the technology is developed, this is the reality. If this is the case, then the most crucial question that must be answered is whether active engagement with visualization technology can effectively boost a learner's information acquisition. Throughout our poll, instructor overhead was identified as the second biggest barrier to visualization technology; yet, this research does not directly address this issue. However, we remain optimistic that educators from diverse backgrounds will begin to develop instructional resources that leverage the aforementioned modes of involvement in the event that the experiment results demonstrate that specific forms of active involvement with visualization yield exceptionally beneficial educational outcomes.

We provide proof for our thesis in the second section by analyzing experimental studies that established best practices for algorithm visualization and showed the effectiveness of visualization. By doing thus, it sets our case against the prevailing beliefs and established industry standards about the use of algorithm visualization. Section 3 offers a taxonomy of potential categories for learner involvement with visualization technologies. The metrics that can be used to assess the learning outcomes that may arise from this kind of engagement are then described in Section 4. We provide efficacy indicators and a methodology for conducting empirical evaluations of algorithm visualization's utility in Section 5. Our engagement taxonomy serves as the foundation for these measurements and approach. The sixth section offers a final discussion of this framework's future prospects. This section also covers future projects and opportunities for collaboration for interested educators in the field of computer science.

Background

Interactive visualisation has been employed in the field of computer science education ever since the 1980s (for example, see [8,10]). Because of the various technical experiences that educators have had over the course of time, a collection of "best practices" has formed over time. In Section 2.1, we present a summary of the findings that are considered to be the most significant. The purpose of a survey that was conducted recently among educators in the field of computers was to determine the prevalent practices, as well as the viewpoints of the instructors regarding the effectiveness and implications of these activities. Section 2.22.2 provides a concise summary of the most important findings revealed by this survey. In Section 2.3, where we present a summary of the inconsistent outcomes of prior experimental investigations on the usefulness of visualisation, we advocate the necessity for a fresh set of tests due to the fact that existing trials have produced inconsistent results.

Overview of Best Practices

1 Pedagogical visualisation incorporates elements from multiple interconnected fields, including typography, algorithms, and psychology. Although many fundamental concepts about style and typography stay unchanged, this adds complexity to the process of distilling the insights gleaned. The guidelines for interactivity, colour and sound design, and display layout are outlined in Khuri's work [35]. While there is no globally known standard for the "commandments of algorithm animation," the following eleven proposals are commonly regarded and have been created based on practical experience: Grant pupils the ability to access materials that aid in understanding the visual depiction. Visualisations, as tangible depictions, possess the capacity to aid learners in comprehending algorithms. Nevertheless, comprehending visualisations can be challenging, and students may struggle to establish a connection between a certain visualisation and the algorithm it is meant to depict. To enhance learners' understanding of the connection between graphical representations and programme elements, there are two possible methods: either dedicate instructional time to reinforce this relationship throughout the course, or incorporate explanatory text or narration into the system's representations.

. 2. Tailor the content to the user's degree of proficiency. Inexperienced learners may feel overwhelmed by an excessive amount of windows or features; they generally prefer to assess an animation using pre-determined input data. On the other hand, advanced learners could gain advantages from additional tools that help them navigate and manage complexity. They would also benefit from the opportunity to create their own input data for more thorough testing of algorithms. Furthermore, animations that are influenced by well-known metaphors, like those seen in comic strips [4], stage plays [21], electronic publications [9], or slide presentations [43], could offer a more straightforward comprehension for beginners. On the other hand, advanced learners may benefit from using tools like BALSA, which provides access to extensive data sets and a wide range of perspectives [8].

3. Present a variety of perspectives. An algorithm can be discerned by analysing the control flow in source code or by examining the state of data structures, among other factors. Exposing the student to several perspectives can enhance their comprehension of the algorithm. Aligning windows with different viewpoints will enable the uniform display of information. Having access to both programme animation views, which highlight and display code during programme execution, and abstract algorithm animation views simultaneously is highly beneficial. By doing so, the student can establish a connection between algorithmic operations and code. An alternative method is to offer pseudo-code instead of real code [57]. In order to provide a sufficient

level of detail, animations of upgraded pseudocode nodes with expand/contract capabilities, which simulate gradual refining, should be appropriately synchronised. From an educational standpoint, it can be beneficial to provide several perspectives in a consecutive manner. The HalVis system [22,23] was designed to demonstrate animation in three distinct stages: an extensive depiction of a function, a detailed animation with multiple synchronised views and explanations, and a widely recognised metaphor to aid learners in understanding the concept.

4. Provide precise details on performance. Efficiency analysis is a crucial component in understanding algorithms. Therefore, integrating the data obtained throughout the execution of the algorithm enhances our overall comprehension of its effectiveness. Simultaneously animating numerous algorithms is an additional approach to strengthen performance data, as exemplified in the film Sorting Out Sorting [2]. The user visually determines multiple rates for the same problem.

5. Present the chronological record of the execution process.

During the algorithm animation, it is typical for the learner to overlook previous steps, misinterpret a step, or have the inclination to see the complete history. Sharing historical facts with the student can help address these issues.

Some algorithm viewpoints include the capacity to include history either explicitly or implicitly. When using JFLAP [29], users have the ability to choose any configuration and observe the path it takes from the beginning state to the specified configuration in a nondeterministic scenario.

6. Encourage adaptable implementation oversight. The versatility of changing the visualisation is evident, including the capability to execute it in both a forward and reverse direction (see references [6, 53, 56] for examples). An efficient and user-friendly visualisation control user interface includes the following buttons: stop, pause, return to the start, go to the end, go back one step, and continuous forward movement [43]. 7. Foster the production of visual aids by students. According to Stasko [55], it is recommended that students create their own visual depictions. By employing this architectural framework, students can acquire understanding of the basic components of the algorithm being studied. Moreover, the act of pupils creating their own artefacts cultivates a heightened sense of accountability within them [24].

8. Authorise the utilisation of personalised input data sets. By allowing students to independently choose the raw data sets they want to use (as shown in [8, 38]), it is possible to increase their involvement in the data visualisation process. The learner is given unrestricted access to study the animation and acquire a deep understanding of how the algorithm works on different types of data. 9. Encourage the development of active investigations. Visualisation systems might employ a "pop quiz" method, where users are frequently given brief questions that require a response, in order to enhance their involvement with the visualisation [22, 44]. Occasionally, it is beneficial to employ two different types of inquiries. Even when given the necessary background, some queries may nevertheless be presented in an unusual sequence. These inquiries aid in understanding development by guiding the learner's attention to certain topics and encouraging self-evaluation. Additional questions may be intentionally included at critical junctures, after which students cannot advance unless they provide accurate answers.

10. Encourage the exchange of participatory feedback. The learners' activities should be supplemented by real-time feedback through a visualisation system. Korhonen and Malmi [37] give an instance of a visualisation system that displays learners with graphical depictions of algorithms and directs them to alter these depictions in order to reproduce the algorithm. Afterwards, the technology offers automated, immediate feedback to students regarding the accuracy of the simulations.

11. Enhance the pictures with relevant anecdotes. Based on educational research rooted in dualcoding theory, the combination of visual aids and explanations is proven to be useful in improving comprehension [41]. Integration could be accomplished by including an explanatory text in a coordinated graphical window or by providing a synced audio track for the visualisation. In addition, one can choose to utilise a conventional method of clarification by consulting textbooks. Bazik et al. [3] highlight the need of having textbooks and animations closely synchronised in order to seamlessly incorporate animations into a course. Due to the lack of a universally effective approach or activity for visualisation, educators must carefully evaluate how to adapt and implement the described concepts. The design of an animation system and its animations should be meticulously prepared, just like any other design project (see to [34] for an example).

Description of the surveys

Prior to the ITiCSE 2002 meeting in Örhus, the Working Group began collaborating and assessing numerous resources. One of the provided items was a synopsis of a manual survey completed by Scott Grissom at the ITiCSE 2000 conference in Helsinki, Finland [19]. The results of Grissom's survey inspired us to carry out a more extensive online poll before the conference. We analysed the replies to Grissom's survey in order to create a framework that would provide guidance for the working group as it carried out its agenda. The working group utilised the SurveySuite tool (http://intercom.virginia.edu/SurveySuite) to create an online survey. This programme simplifies the process of creating surveys for researchers by allowing them to choose items from a predetermined list. 134 Upon obtaining a specific URL, participants proceed to fill out the survey by inputting their answers to the different questions into the form and then hitting a submit button to finalise the survey. After the responses are sent, they are collected and stored in a database. The researcher can access this database to view profiles of the received responses and spreadsheets that show the distribution of the responses for local analysis. The concluding questionnaire before to the conference consisted of four components. The study began by asking participants about their experiences with visualisations. We utilised a range of tools to evaluate individuals' viewpoints and experiences with different types of visualisation. Only a small portion of the evaluation items were specifically created to identify mediated transfer, which is an instructional method where teachers assist students in making connections between visual stimuli and the concepts they are learning. The second component of the pre-conference survey aimed to provide a detailed profile of the teaching situation for each participant. The topics discussed were the count of educators and learners at the establishment, the characteristics and size of courses, the accessibility and arrangement of equipment, and the use of different types of visualisation.

The final portion of the preconference survey requested information from individual respondents regarding their experience as educators, the geographical location of their institutions, and the sources they used to obtain knowledge on visualisation. In addition, we requested the contact information of the participants, along with information about how they were informed about the pre-conference survey. Participants were offered a final chance to submit any further information or reflections in the Closing Thoughts section. Prior to our departure for the ITiCSE 2002 conference in Örhus, we had amassed a total of 29 answers. After a thorough discussion of the upcoming findings, we reached a consensus that it would be advantageous to administer a concise and casual survey to the conference participants using index cards. Our index card survey consisted of only two questions. 1) How often do you integrate algorithm visualisations into your

lessons? The utilisation of visualisations can enhance the comprehension and analysis of computer science. Which of the following options is represented on a five-point scale: Strongly disagree, Agree, Neutral/No Opinion, and Strongly agree? The response options on a four-point scale are as follows: rarely (occurring once or twice per term), occasionally (occurring every other week), and never. The survey collecting index cards were distributed before the conference session began, at the same time as each Working Group presented a brief summary of their efforts. The participants were directed to promptly answer the two questions that were displayed on the front of their index cards. In addition, they were instructed to provide any further comments or inquiries on the back of the card. Before presenting the findings of the preconference survey, we gathered the completed index cards.

Engagement Taxonomy

We describe six broad categories of learner interaction with visualisation technology to help explain how learners participate in an educational setting that uses visualisation. The first category, "No viewing," denotes that no visualisation technology is used at all, since it is undoubtedly possible to learn an algorithm without using it.

1. No watching
2. Seeing
3. Reacting
4. Modifying
5. Building
6. Making a presentation

Viewing

looking can be considered the primary mode of interaction with visualisation technology, as all other modes of interaction essentially require some sort of looking. This concept is depicted in the Venn diagram in Figure 2, where "Viewing" is identified as the overarching category that includes all other types of participation. Furthermore, the act of observing is likely the style of engagement that offers the most diverse range of variants. For example, a student can opt to passively observe an animation, but they can also control its speed and direction, use several windows to see different perspectives, or refer to the accompanying written or spoken explanations. Viewing is inherently a passive kind of engagement. It does not entail any active contact with the visualisation, save for the capacity to change perspectives and control the execution of the visualisation. It is crucial to acknowledge that visualisation includes auralization in its whole [49, 50]. Thus, "hearing" is categorised inside this group.

Responding

"Responding" is the third category in the engagement taxonomy.

Responding to inquiries about the system's visualisation is the main task in this area. For instance, instructors may provide queries to students like, "What will this visualization's next frame look like?" "What source code does this visualisation represent?" (prediction) ï (coding) ï "What is the algorithm's worst- and best-case efficiency as represented by this visualisation?" (Analysis of efficiency) ï "Is there any fault in the method this visualisation represents? (Fixing bugs) During the responding mode of engagement, the student makes use of the visualisation to help them with question answers. As a result, there is not much interaction with the visualisation during the engagement. On the other hand, answering a question might require actions that lead to more viewing activities. For instance, changing the source code and creating a new visualisation

could be a legitimate answer to the query, "Is there a bug in this programme?" during a debugging session.

The category of "responding" is the third classification in the engagement taxonomy. The main duty in this region is to address queries related to the system's visualisation. For instance, educators may ask learners questions like "What will the next frame of this visualisation illustrate?" "Which source code does this visualisation depict?" The division of prediction by coding. "What are the minimum and maximum efficiency values of the algorithm, as depicted in this visualisation?" The analysis of efficiency, denoted by the symbol η, "Does this visualisation exhibit any inherent flaws in its methodology?" (Error resolution) During the interactive learning process, the student utilises a visual assistance to help them react to the questions. As a result, there is very little contact with the visualisation during the engagement. On the other hand, answering a question may require actions that lead to more activities associated to viewing. As an illustration, making changes to the original code and creating a new visualisation could be considered an appropriate answer to the question, "Does this programme have a bug?" during the diagnostic session.

Algorithmics and Data Structures in the Context of Bloom's Taxonomy

We generate illustrative tasks within the domains of algorithms and data structures to demonstrate the process by which an investigator might delineate a specific region according to Bloom's breakdown. We acknowledge the inherent complexity of this mapping endeavour and the potential complications that may arise with the subsequent categorization. A wide array of tasks associated with algorithms, including analysis and implementation, entail remarkably diverse degrees of complexity. Such assignments traverse numerous tiers according to Bloom's taxonomy. Prior to delving into our exhaustive enumeration of knowledge levels, we shall attend to several noteworthy concerns in the subsequent segments of this section. In algorithmics, the varying degrees of complexity of fundamental concepts constitute the initial obstacle. Similar to other disciplines, algorithms encompass a substantial quantity of unfamiliar terminology and concepts that contribute to the field's lexicon. Fundamental notions in graph theory, including nodes, edges, cycles, and routes, are easily understood. On the contrary, concepts such as the depth-first search (DFS) method for traversing graphs are considerably more difficult to implement. The lowest taxonomy level (knowledge level) is characterised by the ability to recognise the names of data structures and algorithms. On the other hand, the ability to explain their operation is classified as the second level (comprehension level). The second disconcerting element is that algorithmics knowledge can be perceived as encompassing both conceptual and practical aspects. It would seem logical to assume that in order to apply an algorithm, a learner must possess a conceptual comprehension of its operation. However, the suitability of employing an algorithm in a programming language for level 2 comprehension or level 3 application is dubious. Undoubtedly, learners demonstrate the application of conceptual knowledge when they execute a conceptually known algorithm. However, implementation serves as an extra and more comprehensive approach to articulating the method. Hence, we propose that level 2 (comprehension) be associated with a programming assignment that demands the student to "write the code for implementing Quicksort," whereas level 3 (application) be associated with a test item that necessitates the student to sort an array of records utilising a sorting algorithm. In practice, it is almost always necessary for students to modify the code examples they have studied in the textbook before applying algorithms to real-world scenarios. Frequently, solutions necessitate the integration of diverse data structures and algorithms. Capabilities required for this

include problem domain analysis, item and structure recognition, selection of appropriate representations for structures, and determination of the most applicable algorithms and structures for problem resolution. Problem analysis is a component of Level 4 (analysis) duties. Construction of the solution corresponds to level 3 (application) if learners are capable of handling it through the use of familiar algorithms, and level 5 (synthesis) if they are required to generate something from the ground up. Distinguishing between different levels of difficulty is an inevitable occurrence; however, assignments requiring the development of novel algorithms must be no less than level 5 synthesis. Learners ultimately attain level 6 (evaluation), which corresponds to the pinnacle of Bloom's taxonomy, when they are obligated to appraise their own solutions based on specific criteria. The third challenge at hand pertains to the intricacy of algorithm analysis, which encompasses elements classified under various taxonomy levels as per Bloom's hierarchy. Level 1 (knowledge) consists of understanding fundamental concepts such as big-O notation and worst-case complexity, while level 2 (comprehension) entails replicating and following the analysis of an algorithm. Students must employ what they have learned regarding algorithm analysis to a specific problem in order to conduct an analysis. However, de to the considerable degree of difficulty exhibited by these analysis issues, it is difficult to assign them to a single level of Bloom's hierarchy. We suggest the following division of this matter into three tiers: level three for application, level four for analysis, and level five for synthesis. Assessors might be required to conduct an analysis of a foundational algorithm that was deliberated upon in the course or to develop a rudimentary iteration of the algorithm. To undertake an analysis of progressively intricate and challenging responses, pupils are required to deconstruct the problem into more manageable tasks (a level 4 analysis exercise). Once these simpler assignments have been distributed, students are free to assess each one independently. Complex problems may require students to employ and combine multiple approaches in order to arrive at a viable resolution. This requires the utilisation of skills associated with algorithmic research, specifically synthesis at level 5. Students have completed level 6 (evaluation) work when they are required to provide a critical assessment of their own analysis methods and results.

Other Factors to be Measured

We discussed the learner's potential knowledge levels in the preceding section. This section delves deeper into the topics that are measurable in terms of learner improvement and the variables that may cause results to be misleading.

1. The advancement of the learner Over the course of a study programme, students should advance to work at more advanced levels as their knowledge increases along Bloom's taxonomy/hierarchy. But this development is not linear. Students can do well or poorly on each level, but the more information they have, the better they should do on the lesser levels. The hierarchical structure of knowledge must be taken into account when evaluating learners' knowledge.
2. Let's say the teacher assigns a task that assesses students' understanding at a certain level and grades it according to a conventional system that uses a point system ranging from 0 to 6. The evaluation research might thus yield outcomes similar to "We noticed that students using method A received an average of 4.6 out of 6 points on an assignment assessing level 2 knowledge, while students using method B received only 3.2 out of 6 points." To ascertain whether this difference is statistically significant, a t-test could be employed.

2. The rate of dropouts Some students choose to leave a course as a result of the challenges they face in their studies.
Such a decision may be made for a variety of reasons. The motivation of the students to study the subject and their attitudes towards the various teaching strategies and resources employed in the course are the most pertinent factors in the context of this report. Since some students may register for the course before they decide to take it, measuring dropout rates is not always simple. Therefore, a more accurate measure of initial enrollment may be the number of students who stick with the course long enough to turn in at least the first assignment or exam for assessment. The regulations set forth by the institution for taking the final exam should also be taken into account when calculating the drop-out rate. Under certain university regulations, students may be able to take the final exam in one or more ways. A good place to start tracking the drop-out rate would be just after the first exam, or right after all exams have been finished. When designing the experiment, it is necessary to decide on the definition of drop-out that will be applied in a particular situation.
3. Time spent learningFor each student to reach the same level of understanding, different learning times are required. The amount of time spent learning can also vary depending on the strategy used. If the instructor wants to cover additional material in the course, this may be crucial. Thus, rather than assigning tasks with a deadline and determining the learner's Bloom's Taxonomy Level,
What Is Possible For The Learner At This Stage Example Assignments And Tasks
4-Analysis ● Recognises how the method relates to other algorithms that address similar or related challenges.
● Is aware of the algorithm's code's invariants.
● Possess the ability to defend, explain, and/or demonstrate the algorithm's correctness.
● The ability to break down a complex issue into smaller, more manageable issues by identifying key components.
● Sort different types of tree structures.
● Examine how Quicksort and Heapsort perform in comparison. ·Explain the workings of Dijkstra's algorithm.
● Describe why Dijkstra's algorithm fails for networks with negative-weighted edges, yet Prim's algorithm succeeds in such cases.
Examine the types of data formats and algorithms required for the development of a search engine.
5-Compositing ● Create solutions for challenging issues requiring a variety of data structures, algorithms, and approaches.
● Examine how effective intricately connected structures are.
● Establish standards for contrasting different approaches.
● Create a search engine and assess how well it uses time and space.
● Create the algorithms and data structures required for a vehicle navigation system.
● Establish a testing environment to evaluate different search structures in a hierarchical memory.
6. Assessment ● Make a case for the modification or combination of one algorithm with another in order to address a new, more difficult problem more effectively
.● Talk about the benefits and drawbacks of various algorithms for the same or related problems.
● Complete an analysis or design evaluation.
● Specify suitable standards for judging the suitability of search algorithms and make the case for their significance.

● Examine hashing and balanced trees in terms of how they approach dictionary implementation.
● Talk about a solution's design and make the case for why it works better or worse than an alternative.
● Talk on how the analysis in the original text could be improved. 147 enhancement, the teacher might provide tasks with an infinite amount of time and track how long it takes for pupils to do each one. One could consider this a successful outcome if employing visualisation inspired students to dedicate more time to their work.
4. Contentment of the learner
There are various reasons why students enrol in a course. Furthermore, they may become less motivated as the course progresses. Therefore, it becomes sense to request comments so that the instructor may find out what the students think of the course. These inquiries may address attitudes towards the topic matter as well as the students' perceptions of the different teaching strategies and resources used in the course.
Procedure
The objective of the technique is to establish the experiment. The experimental procedure consists of three main stages: pre-test, task completion using materials, and post-test. This tripartite procedure does not encompass every aspect of utilising the data collection and visualisation materials. We delegate the task of process design to the instructor, allowing them to tailor it according to their schedule and the needs of their pupils. To ensure reliable findings, it is recommended to desig the experiment by dividing the students into two or more randomised groups. These groups should have comparable population distributions based on the specified covariants outlined in Section 4.4. Nevertheless, due to the frequent impracticality of such groupings, an alternative approach could be utilised, such as the following:
● If a course is provided by an institution with two parts, each section may utilise a singular form of involvement, which can then be compared. ● In some educational institutions, where students should receive identical instruction, a specific approach may be implemented in one section of the course during the first half of the semester, but not in the other. Subsequently, both segments undergo the post-test. In order to ensure equitable learning opportunities, the group that did not participate can be treated in the same way after the post-test, once the experimental data has been collected.
● If the same course is taught over two semesters, it is possible to use one teaching approach during one semester and a different one during the other.

Conclusion

With the use of this report, computer science educators will be able to assess the correlation between a learner's type of engagement with a visualisation and the many categories of understanding that are impacted by that engagement. To aid in characterising the type of engagement employed in these kinds of studies, we have established an engagement taxonomy. We have also discussed the ways in which different fields of computer science might employ Bloom's taxonomy to distinguish between different kinds of understanding. We have provided a framework for performing experiments that employ these two taxonomies to determine the independent and dependent variables, respectively, based on these taxonomies. We want to create a number more focused experiments based on this concept in the upcoming year. The scope of these trials will enable researchers from many universities to work together. Teachers who would like to participate are encouraged to get in touch with either of the co-chairs of the working group at naps@uwosh.edu or roessling@acm.org.

Reference

Andres, H. P. (2017). Active teaching to manage course difficulty and learning motivation. Journal of Further and Higher Education, 43(2), 1-16. https://doi.org/10.1080/0309877X.2017.1357073

Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in introductory programming. ACM SIGCSE Bulletin, 39(2), 32-36. https://doi.org/10.1145/1272848.1272879

Bergmann, J., & Sams, A. (2012). Flip your classroom: Reach every student in every class every day. International Society for Technology in Education. https://www.rcboe.org/cms/lib/GA01903614/Centricity/Domain/15451/Flip_Your_Classroom.pdf

Bonwell, C., & Eison, J. (1991). Active learning: Creating excitement in the classroom. ASHE-ERIC Higher Education Reports. The George Washington University.

Borges, R. P., Oliveira, P. R. F., da R. Lima, R. G., & de Lima, R. W. (2018). A systematic review of literature on methodologies, practices, and tools for programming teaching. IEEE Latin America Transactions, 16(5), 1468-1475. https://doi.org/10.1109/TLA.2018.8408443

Bosse, Y., & Gerosa, M. A. (2016). Why is programming so difficult to learn? Patterns of difficulties related to programming language. ACM SIGSOFT Software Engineering Notes, 41(6), 1-6. https://doi.org/10.1145/3011286.3011301

Cavanagh, M. (2011). Students' experiences of active engagement through cooperative learning activities in lectures. Active Learning in Higher Education, 12(1), 23-33. https://doi.org/10.1177/1469787410387724

Duffany, J. (2015, July). Active learning applied to introductory programming. Proceedings of the 13th Latin American and Caribbean Conference for Engineering and Technology: Engineering Education Facing the Grand Challenges What Are We Doing? Santo Domingo, Dominican Republic. https://doi.org/10.18687/LACCEI2015.1.1.246

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. Information and Software Technology, 50(9-10), 833-859. https://doi.org/10.1016/j.infsof.2008.01.006

Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. Proceedings of the National Academy of Sciences, 111(23), 8410-8415. https://doi.org/10.1073/pnas.1319030111

Giannakos, M. N., Krogstie, J., & Chrisochoides, N. (2014, November). Reviewing the flipped classroom research: Reflections for computer science education. Proceedings of the 2014 Computer Science Education Research

Conference, 23-29. https://doi.org/10.1145/2691352.2691354

Gomes, A., & Mendes, A. J. (2007, September). Learning to program-difficulties and solutions. International Conference on Engineering Education, Coimbra, Portugal. https://www.researchgate.net/publication/228328491_Learning_to_program_-_difficulties_and_solutions

Gomes, A., & Mendes, A. (2014, October). A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations. Proceedings of the IEEE Frontiers in Education Conference, 1-8.

Berlin, Germany. https://doi.org/10.1109/FIE.2014.7044086

Kitchenham, B. A. (2012, September). Systematic review in software engineering: Where we are and where we should be going. Proceedings of the 2nd international Workshop on Evidential Assessment of Software Technologies.
Lund, Sweden. Association for Computing Machinery. https://doi.org/10.1145/2372233.2372235
Koulouri, T., Lauria, S., & Macredie, R. D. (2015). Teaching introductory programming: A quantitative evaluation of different approaches. ACM Transactions on Computing Education, 14(4), 1-28.
https://doi.org/10.1145/2662412
Lage, M. J., Platt, G. J., & Treglia, M. (2000). Inverting the classroom: A gateway to creating an inclusive learning environment. The Journal of Economic Education, 31(1), 30-43. https://doi.org/10.1080/00220480009596759
Luxton-Reilly, A., Sheard, J., Szabo, C., Simon, Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., & Scott, M. J. (2018, July). Introductory programming: A systematic literature review. Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, 55-106. Larnaca, Cyprus. https://doi.org/10.1145/3293881.3295779
Martins, V. F., Concilio, I. de A. S., & Guimarães, M. de P. (2018). Problem based learning associated to the development of games for programming teaching. Computer Applications in Engineering Education, 26(5), 1577-
1589. https://doi.org/10.1002/cae.21968
Mazur, E. (1997). Peer instruction: A user's manual. Prentice Hall. https://doi.org/10.1063/1.881735
Mazur, E, & Somers, M. D. (1999). Peer instruction: A user's manual. American Journal of Physics, 67(4), 359-360.https://doi.org/10.1119/1.19265
McGettrick, A., Boyle, R., Ibbett, R., Lloyd, J., Lovegrove, G., & Mander, K. (2005). Grand challenges in computing: Education – A summary. The Computer Journal, 48(1), 42-48.https://doi.org/10.1093/comjnl/bxh064
Medeiros, R. P., Ramalho, G. L., & Falcao, T. P. (2019). A systematic literature review on teaching and learning introductory programming in higher education. IEEE Transactions on Education, 62(2), 77-90.
https://doi.org/10.1109/TE.2018.2864133
Michael, J. (2006). Where's the evidence that active learning works? Advances in Physiology Education, 30(4), 159-167. https://doi.org/10.1152/advan.00053.2006
Noordin, K., Nasir, A. N. M., Ali, D. F., & Nordin, M. S. (2011). Problem-Based Learning (PBL) and Project-Based Learning (PjBL) in engineering education: A comparison. Proceedings of the IETEC'11 Conference,Kuala Lumpur, Malaysia.
O'Flaherty, J., & Phillips, C. (2015). The use of flipped classrooms in higher education: A scoping review. Internet and Higher Education, 25, 85-95. https://doi.org/10.1016/j.iheduc.2015.02.002
Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., & Paterson, J. (2007). A survey of literature on the teaching of introductory programming. SIGCSE Bulletin, 39(4), 204-223.
https://doi.org/10.1145/1345443.1345441

Peng, J., Wang, M., & Sampson, D. (2017, July). Scaffolding project-based learning of computer programming in an online learning environment. IEEE 17th International Conference on Advanced Learning Technologies, 315-
319. https://doi.org/10.1109/ICALT.2017.17
Prince, M. (2004). Does active learning work? A review of the research. Journal of Engineering Education, 93(3),223-231. https://doi.org/10.1002/j.2168-9830.2004.tb00809.x
Qian, Y., & Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. ACM Transactions on Computing Education, 18(1), 1-24. https://doi.org/10.1145/3077618
Richardson, D. (2008). Don't dump the didactic lecture; Fix it. Advances in Physiology Education, 32(1), 23-24. https://doi.org/10.1152/advan.00048.2007
Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion.Computer Science Education, 13(2), 137-172. https://doi.org/10.1076/csed.13.2.137.14200
Simon, Luxton-Reilly, A., Ajanovski, V. V., Fouh, E., Gonsalvez, C., Leinonen, J., Parkinson, J., Poole, M., & Thota, N. (2019, December). Pass rates in introductory programming and in other STEM disciplines. Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education, 53-71. Aberdeen, Scotland. https://doi.org/10.1145/3344429.3372502
Sleeman, D. (1986). The challenges of teaching computer programming. Communications of the ACM, 29(9), 840-841. https://doi.org/10.1145/6592.214913
Vihavainen, A., Airaksinen, J., & Watson, C. (2014, July). A systematic review of approaches for teaching introductory programming and their influence on success. Proceedings of the Tenth Annual Conference on International
Computing Education Research, 19-26. Glasgow, Scotland. https://doi.org/10.1145/2632320.2632349
Wang, H.-Y., Huang, I., & Hwang, G.-J. (2016). Comparison of the effects of project-based computer programming activities between mathematics-gifted students and average students. Journal of Computers in Education, 3(1), 33-45. https://doi.org/10.1007/s40692-015-0047-9
Watson, C., & Li, F. W. B. (2014, June). Failure rates in introductory programming revisited. Proceedings of the
2014 Conference on Innovation & Technology in Computer Science Education, Uppsala, Sweden, 39-44. https://doi.org/10.1145/2591708.2591749