The Role of Observation in Computer Science Learning

Rob Faux
Computer Science Education, Distance Education and Curriculum Consultant
rfaux@oneota.net

Abstract:

This paper investigates how observation is critical to learning and provides strategies for supporting observation in Computer Science learning. Observation is a combination of using the senses to experience an actual event followed by reflection of what was experienced. Frequently, observation is not supported when course structures fail to encourage reflection that leads to long-term integration of new knowledge. There are many opportunities in the traditional Computer Science classroom to rectify this imbalance. A set of possible approaches are discussed in this document.

Keywords: observation in learning, computer science, teaching techniques

1. Approaches to Learning

Just as there are many different kinds of people in the world, there are many different approaches to learning. Teachers at every level need to recognize the personal nature of learning and should work to support a broad variety of approaches to learning. The application of different teaching methods to support the various learning styles should be a goal of every educator at every level for any subject. With intelligent use of various facilitation techniques, instructors may reach a larger and more diverse group of learners.

The focus of this paper is on the facilitation of learner observation to support learning. Observation can certainly be a part of learning regardless of the teaching techniques used and the learning preferences exhibited by the student. However, it is the author's opinion that educators need to recognize the power of observation in learning and work to facilitate its use in the classroom.

Passive versus Active Learning

The current trend in educational circles is to support active learning techniques, such as collaboration and 'hands-on' approaches. It is argued that action on the part of the learner will increase their interaction with the subject matter and aid them by forcing some investment in their own learning [3]. In Computer Science courses, it is common to encounter collaboration in the form of team projects (especially in later courses) and various hands-on approaches in labs and via programming and other projects are frequently used. Other active learning approaches, such as service learning and discussion methods are much less common in CS, but are nonetheless potentially useful tools for the CS educator. Innovation in the facilitation of active learning is currently a popular field in educational research and is certainly worthy of integration into the repertoire of any CS faculty person's teaching toolbox.

Passive learning, on the other hand, is quite often seen as the reading and lecture portion of classes. Both of these methods are intended to impart information to learners and rote memorization is frequently a necessary component [2]. Many online tools using multimedia are also highly passive in approach since many consist largely of reading text, viewing pictures, or listening to audio components. Computer Science faculty are frequently at the forefront of those using online tools for class notes, tutorials and other items to support learning.

Unfortunately, the nearly exclusive use of reading and lecture by many instructors in the past has brought about a backlash against passive learning methods. This often leads to attempts by some facilitators to completely forsake these tools, which certainly is no better than avoiding active learning approaches [5]. In other cases, it causes instructors to be defensive about their use of passive approaches and leads to a polarization between faculty that does not benefit the learner. Certainly, there are advantages and disadvantages to each teaching technique. It is also quite possible for an instructor to be effective or ineffective using any given strategy. Rather than discounting a technique or approach entirely, instructors should concern themselves with the relative benefits of various teaching approaches. Additionally, teachers should continuously challenge themselves to improve the variety and quality of their instructional methods.

Concrete versus Abstract Capabilities

Most learners who are traditional college ages (17-23) are entering a point in the maturation process where they can begin to process and learn from abstractions [8]. Of course, each individual will be at a different point in this development. In some cases, an individual may be quite willing and able to accept general assumptions, models and axioms and apply them successfully. However, most individuals will struggle to some extent unless there are concrete examples and situations from which they can learn. Even given concrete examples, some people may find it nearly impossible to generalize the concepts and apply them to different concrete examples. By the same token, there are individuals who can readily accept high-level concepts, but will struggle to apply them in a precise situation.

The facilitator of learning must recognize the difficulties some may have in crossing between abstractions and applications. There must be recognition by Computer Science educators that models, code fragments, and computing languages can be very abstract from the perspective of the learner. Similarly, full programs for specific tasks or scenarios can provide concrete examples. To make matters more complex, learners (and instructors) do not always recognize the difference between an abstraction and an application. This becomes clear when learners complain that a test question was unfair, even though the instructor feels it is actually a similar problem to one that was covered extensively in the classroom. Faculty must work to point out the difference between a specific case and a general case. Once this is done, they can facilitate the translation from one to the other.

What do we Mean by Observation?

Taken literally, observation can certainly occur regardless of the level of activity or the level of abstraction in a learning event. Certainly, one could argue that observation is a part of any learning. After all, a lab situation requires the learner to view the problem, interpret it, attempt to solve it, and then see how the solution fairs. The observation of the success or failure of the solution is part of the key to learning concepts in a laboratory environment. Similarly, a student in a lecture hall observes by listening to the instructor, viewing the notes and other visual aides and participating in discussion. These scenarios certainly seem to fit a traditional sense of what it is to ‘observe.’ However, if we leave our definition of observation here, we leave out a critical part of the learning process.

In each of the cases above, observation should be a 'two-part adventure.' The first part of the process involves using the senses to watch what is happening and experience the event. As was mentioned above, many of us might consider this to be the full extent of observation in learning. However, observation for learning includes a second step. The learner needs to reflect on the experience and determine where the experience belongs in their pool of knowledge. Thus, in order to facilitate observation in learning, we must consider ways that will both provide a useful experience and that will promote reflection and integration of that experience [1,2,5].

With this definition in mind, it becomes fairly obvious that facilitation of observation will usually require more concrete experiences than abstractions. This does not mean that abstraction is ignored. As a matter of fact, concepts, applications, and instances of traversals between the two can often be successfully 'observed.' But, we must recognize that these are still instances of the more specific models. It is hoped, by the use of applications in the experience, that the learner will be able to integrate the concept (or the abstraction) as they reflect on what they have experienced. Further, events that show the process and steps from concept to application (and/or vice versa) can facilitate successful reflection.

Observation should occur regardless of the level of 'activity' in a learning event. However, there are good reasons to consider observation as a passive activity. Frequently, a highly active situation will cause the learner to fail to see what needs to be seen during the event. Continued activity tends to dull the willingness to process and reflect on what has been happening. In other words, the activity becomes the focus, rather than the learning. The result is that participants become trained in a skill, but have insufficient understanding of the reasons behind the process. This makes it increasingly difficult for the individual to handle abstractions and exceptions. Teachers need to work to encourage moments of observation in every active learning event so that the activity might lead to more complete understandings of what has occurred.

On the other hand, passive events, such as lectures, can certainly be endured with a minimum of observation. In these cases, some level of 'activity' can promote the necessary reflection and integration. For example, instructors who provide learners with questions prior to a lecture may find that learners are more willing to pay more to the content of the lecture. Regardless of the level of activity in the event, the instructor must identify ways that they can facilitate the necessary observation that will lead to learning.

A Revelation Regarding Observation

The 'trial and error' methodology is very commonly accepted in most European-style educational format. Many persons who claim to be kinesthetic (hands-on) learners who have grown up in this system see 'trial and error' as being the most common approach. In this instance, one tends to learn by making mistakes and seeing how these mistakes bring about results that are not necessarily those that were anticipated. With this approach, better learning occurs when time is taken to view the results and reflect on how those results differ from what was expected. This effort brings the learner to a point where they can make conjectures as to why things were different and how changes might bring about the desired result.

While 'trial and error' certainly allows for both phases in observation, it does not necessarily mean that every learner will take the time for the entire process. Frequently learners simply get frustrated with the results. When this occurs they either make 'knee-jerk' alterations to the solution or they seek a 'ready-made' answer (often supplied by an instructor). In short, 'trial and error' methods tend to encourage action, but the focus on observation is often neglected.

Some interesting reading regarding the traditional learning approaches of many Native American peoples reveals a different approach to learning that might provide an interesting balance to 'trial and error' tactics. In this scenario, the teacher demonstrates how something can be successfully completed while the learner(s) serve as an audience. After observing a number of repetitions, the learner reflects on what they have experienced and attempts to perform the task on their own (usually with no audience) [4]. Perhaps there will be some 'trial and error' at this stage, but the learner does have a working model as a basis for their efforts. The advantage of this approach is the early focus on the process of observation.

Certainly this approach could be useful in demonstrating both successful and unsuccessful concepts and approaches in the Computer Science classroom. At the least, this could encourage and exercise the observation skills of the participants. Further, integration of this approach with 'trial and error' methods should provide opportunities for individuals with varying learning styles to succeed.

The Value of Observation in Learning

Rather than provide lengthy descriptions of the advantages for two-part observation in learning, it is appropriate to simply provide a list. This gives more opportunity to discuss specific strategies in the CS classroom rather than a larger focus on theory. As the reader views the strategies for facilitating observation, it may be beneficial to reconsider the following as they relate to the technique. It is doubtful that many will disagree that each of these can be true on a philosophical level. However, consider whether or not events in the classroom, as they stand now, fully support these statements.

Valuable learning occurs:

- in seeing and reflecting on (observing) successful processes that are modeled by others.
- in observing failed processes.
- in observing on the response to a failed process.
- in observing completeness.
- in observing repetitions.
- in reflection on one's actions or activity.
- in observing comparisons.
- in observing decision making processes within a larger concept or process.
- in observing the details of a process or situation.
- in observing the impact of a process or event.
- in observing the tools of the trade as they are used.

2. Teaching Approaches that Facilitate Observational Learning

Many techniques facilitate observational learning in Computer Science. The following examples are only a small group of ideas used by the author in CS classes and are presented here in the hope that others may benefit in their use. Some of these ideas have been frequently used by CS/IS faculty for some time, others seem to be common sense and others are a reflection of the instructor himself. Of course, each of these examples will have varying levels of success depending on the learners, the instructor and the environment. Further, faculty must consider the time and effort required (for themselves and the students) in order to make these events work. Before rejecting any of these approaches on that basis, one must consider the potential value an event such as these might have versus those events that are already planned.

Full Program Examples

Early in Computer Science, learners are working to learn a computing language. Frequently, the only full program examples seen by the learner is an initial 'Hello World' example in one of the first classes and whatever examples exist in the textbook. It is natural for instructors and authors to provide code fragments, rather than complete code in order to provide examples for learners. After all, it is tedious to write program headers and the like when they are far from the point to be made. However, we should not forget the value of viewing success and of viewing completeness as a part of observation. The successful authoring of a complete, working program by the instructor while learners view the process can have a strong impact on many of those learning programming languages.

When providing full program examples, the instructor should consider that simply providing copies of a full program does not suffice. Learners at the post-secondary level often have good intentions (and sometimes they don't) of reading materials provided for them, but these materials are infrequently perused. More importantly, providing a fully written and completed program deprives the learner of the value of viewing the process. There is great strength in combining process, completeness and repetition, which is provided by full program examples.

Before dismissing this idea as being too time-intensive or as being unnecessarily redundant, one must consider the audience. First, learners do not have the wealth of experience that the instructor presumably has. Things that the teacher takes for granted aren't necessarily even recognized as important by the newcomer. Second, judicious use of full-program examples can provide the opportunity to impart new concepts as well as repetitions for already covered, yet still uncertain, ones. And, finally, the ultimate goal should be to bring the students to an appropriate level of understanding of the topic. Rote memorization of concepts, algorithms and patterns certainly may give the appearance of understanding, but it is not a wholly accurate indicator.

Provision of Working Solutions to Assignments

Projects, assignments and even exams are often viewed by professors as activities with a definite 'end'. Unfortunately, the result is that there is a focus on the initial activity, but no focus is given to the learning opportunity that comes after assessment or completion of the event. This effectively eliminates the second part of the observation process that is so critical to long-term retention of concepts. For example, the student works to memorize for the exam, performs their 'regurgitation' for the exam, and receives a score two weeks after that point. No further discussion or effort is expended (in most cases) on these questions or topics so that a better understanding can be reached (unless they hope to cajole the instructor into a higher score). A large part of this attitude comes from the feeling brought to the classroom that one must always be moving forward in the topic material. But, how can learners be expected to move forward when key concepts are misunderstood?

An excellent method for encouraging observation is to provide learners with solutions to exercises, projects and exams. Once again, we must recognize that reading handout material may not be the best solution. It is often more beneficial to start the process in the classroom with a period of time to review the event. The instructor usually is aware of those concepts and issues that caused the most trouble and can isolate those most likely to help a majority of the students. Further, identification of the most common mistakes can allow the teacher to illustrate the error and provide solutions that work by modifying the solution that was in error. This provides the learner with an opportunity to view the process of turning a 'failure' into a 'success.' Certainly, this is a valuable lesson for any person involved in computing.

There are, of course, some issues that would cause instructors to choose to avoid this approach. First, there is the assumption that evaluation can occur quickly so that assessment information can be used to select appropriate subjects to discuss. This may not necessarily have to be the case, however, since experienced instructors often have a sense for what has caused problems in the past. Second, there are problems with providing full answers to students. These assignments can’t readily be reused in future classes. However, teachers need to remember that the primary goal is successful learning, rather than easy grading or assignment development. The difficulties brought about by providing an opportunity to reflect on graded events are actually quite small as compared to the potential learning value.

'Playing Computer'

The event of fast compilers and near immediate feedback regarding a program’s workability has removed some of the motivation to walk through a solution prior to implementing. When CPU time cost the student money, learners found themselves spending a great deal of time tracing and re-tracing their solutions prior to submitting them for compilation and running. While it is certainly beneficial that learners have unlimited CPU time and speedy compiling and testing tools, they should still be given the opportunity and motivation to see and use tracing and walkthroughs of algorithms and code.

An instructor can provide opportunities to illustrate the workings of a solution by walking through solutions and code manually. They can supplement this approach by encouraging and demonstrating the use of debuggers to follow code as it executes. And, while this certainly sounds like a tool for earlier courses in a CS program, the instructor should also model its use in later courses to illustrate the power code and solution tracing can have. This approach provides learners with a demonstration of a strategy or process for working through computing problems that can be quite valuable.

Debugging

The process of debugging is usually one that is taught nearly entirely by ‘trial and error’ processes in laboratories and in projects. Perhaps learners are given a quick tour of the programming environment at the beginning of their first class, or they are provided with an introduction to basic compiler errors. However, learners are often more successful in debugging if they are given the chance to see successful debugging strategies in action. Usually, this process has greater impact if it occurs after students have had a chance to experience the frustration of debugging themselves. Because they now have a good reason to seek out successful strategies, they will view and integrate them quickly as they are modeled by the instructor.

An excellent method of promoting debugging strategies early in the CS program is to gather code written by learners that does not compile or run correctly. These files can be used to demonstrate how one can look for problems in the code. This approach increases the feeling of ownership in the learners and may increase their desire to observe successful techniques. Later in CS programs, we should not ignore chances to share or reiterate useful debugging approaches. Instructors must remember that learners have to be at a point where they are ready to learn new concepts. Frequently, learners simply are not ready the first time it is presented. A second presentation, at a later point in time, my find them quite ready to accept and integrate what they experience. In other words, instructors should not view debugging as a ‘unit’ to be covered. Instead, debugging should be modeled, discussed and viewed throughout larger topics.

This approach certainly costs the instructor class time and preparation time. However, consider the large amount of time commonly devoted to individual debugging sessions in nearly every CS class. It stands to reason that faculty will benefit from a concerted effort to make debugging techniques part of their focus. This should encourage observation from the standpoint that it supports the process of experiencing and integrating tools for us in Computer Science. This approach models both process and success in response to failure.

Developing and Implementing Test Plans

As an instructor, we certainly speak of testing code and verbally encourage it in our students. However, testing is frequently viewed as tiresome, irritating and less creative than the more development oriented portions of our projects. As a result, instructors and students tend to do very little real testing. Certainly, faculty members are well aware of what a good set of tests might be for a given piece of code. But we forget that learners do not have this experience. Naturally, they will consider one set of test data to be sufficient. If it runs successfully, they feel that the project is complete and no more effort is required.

Testing and planning for testing can be one of the areas in CS that strongly supports observation for learning. The testing process provides the learner with a chance to consider what it is that will determine if a solution is working or not. They must take the time to reflect on the problem and its limitations, so that they can select appropriate tests. Further, they are given the chance to move away from active development so that they can observe how their solution interacts with the data given. Once they have viewed the results, they must determine whether the results are appropriate and how things should be altered to get better results. This is the step in ‘trial and error’ processes that is necessary in order to encourage two-part observation.

There are a number of approaches that can be used to encourage testing and test plans. First, the instructor should model testing for learners frequently. If the professor is illustrating a new algorithm for the class, he/she should take the time to demonstrate how the algorithm performs in boundary cases (for example). If demonstration code is provided, time should be taken to walk through a test set to show correctness (and maybe even failure) of the code. At the very least, a test plan, test suite or set of tests might be provided so that learners can be exposed to them. Learners can then be asked to determine why each test was included in the plan. This encourages actual reflection and integration rather than blind acceptance of listed procedures.

Learners can also be encouraged to test by requiring a test plan and set of test results for each project and/or laboratory exercise. By making the process a required part of the event, students become familiar with its inclusion. Further, inclusion of these test sets can also provide the instructor with more resources for evaluating the work of the learner.

Making Mistakes Into a Valuable Resource

Professors should be viewed as being competent in their field. However, we should not make the mistake of perpetuating the myth that we are infallible. Instructors who make mistakes in the classroom should not work to cover them up. Instead, they should recognize the value in turning the situation into a learning opportunity. It is not suggested here that faculty should make mistakes continuously. After all, it is valuable for a learner to observe success. Instead, the suggestion is that failure, and our reaction to it, can be every bit as useful for the learner to observe as success.

Providing oneself with the opportunity to make mistakes may sound a little ridiculous and even a bit scary. However, we should consider that professors certainly should be comfortable with their topic and they should be quite capable of responding to and correcting errors. Rather than entering the classroom with everything worked out prior to the event, there are times when it is more useful to enter the classroom with only the problem in mind, but no solution. The process of working to a solution can then be displayed for the learners in such a way that they can observe how the instructor handles adversity in the process. As the teacher works through the problem, they should clearly demonstrate what they are doing and why they are doing it. If a problem is encountered, they should indicate how they discovered the problem and why it is a problem. At that point, they can show how they will deal with the situation.

This approach is often very enlightening to both faculty and students. Not only does it provide learners with a chance to observe the processes used by an experienced individual, it also gives the instructor a chance to consider the problems of a student. Learners remain under the shadow of constant evaluation of their work and we often fail to recognize the very real stress this puts on the individual. On the other hand, performing work in front of a live audience can certainly remind one of the stress of problem solving for the benefit of others!

Another valid approach is to purposefully make mistakes so that the error can be demonstrated and solutions presented. In this case, it is usually best to be clear that you are intending to make mistakes and that the focus is on what can go wrong and how it can be corrected. If one includes an intentional mistake without forewarning the students, it will often backfire since students become unsure as to the agenda being pursued by the instructor. Announcing an intentional error after the fact is rarely taken well. On the other hand, unintentional errors provide learners with a chance to see how mistakes can be made into useful learning experiences.

Peer Interaction

Interaction and collaboration are excellent methods of supporting observation in learning. The simple camaraderie felt by those placed in the situation of dealing with a common problem tends to promote sharing of perspectives and opinions that can lead to a better understanding of the task and the concepts behind it. The effort of successfully conversing about the topic requires the participant to sharpen their own observation skills. Sometimes one learner can model or tutor for another student. In other situations, the learners use team efforts to attempt the task at hand. Regardless of the situation, the interaction encourages observation because communication requires that the students attain certain levels of understanding in order to converse on the problem topic. In short, peer interaction supports observation by giving learners an ulterior motive to incorporate observation into their learning.

Instructors can support peer interaction by providing paired projects and team projects. Similarly, teachers can encourage and facilitate peer tutoring and study groups. By encouraging students to accept learning efforts as a part of a community effort, it is possible that individuals will begin to accept that observation and learning can occur practically anywhere at any time.

Stump the Teacher

Providing a new twist to the classroom usually gets the attention of students. In this case, the instructor challenges the students to find questions about the subject that might stump him or her. For example, a second term programming class was challenged to come up with a single-task recursion problem that the instructor could not solve in the class. In this case, the class was given a day to try to come up with problems (either individually or as a group). The day of the challenge was extremely successful as the teacher worked with these problems while the learners observed the process. Learners were asked to take time after the class to write down strategies used by the teacher in solving the problems presented. Many learners expressed favorable opinions of this event.

Obviously, this event requires that learners provide sufficient problems for this to work. Further, there are certainly individuals who will hunt down unsolvable or inappropriate problems for the class. Even so, the instructor must be prepared to discuss how the problem can be approached and why it can’t be done in entirely in that class. The second difficulty is that the instructor places his or her ego on the line and surrenders some control of the classroom to the students. The teacher must be ready to admit defeat while illustrating approaches to solving the problem. Further, it is important that careful guidelines for problem selection be given so that there is a higher likelihood that the event will succeed.

Open Ended Questions

Computer Science, with its strong mathematical background, tends to often lend itself to short, close-ended questions. Program syntax is either right or it is wrong. Algorithms either find the correct solution or they do not. The more heavily mathematical background of many faculty tends to make them much more comfortable with these types of questions. Further, it is far easier to assess the response to such questions than it is to assess responses to an open-ended question. However, there is also a great deal of flexibility within the structure of Computer Science. An algorithm may be written more than one way in order to produce correct answers. Databases may be optimized different ways in order to promote efficiency. Large project designs may be implemented any number of ways, each having different sets of pros and cons. In other words, there is a need for the CS student to develop skills in determining which solutions are best for a given situation. These skills lend themselves to open-ended questions.

The open-ended question also promotes observation in learning. If a learner is asked to optimize a database using a prescribed set of steps with very few options, they have no reason to really observe what is happening. Instead, they simply look at the 'cook book' and follow the 'recipe' in hopes that the correct solution will appear at the end. On the other hand, learners could be asked to optimize the same database and explain WHY they think it is now optimized. Or, if they were encouraged to give reasons for the changes they make, they are more likely to 'step back' from the process and reflect and integrate this knowledge. Once the 'why' is merged with the 'how', both become easier to remember and the task is more likely to be performed adequately in the future.

Another use of open-ended questions is to supply questions at the beginning of a lecture section that ask students to formulate opinions about topics covered in the lecture. For example, a class covering various shortest path algorithms may include questions centering around when one might select one algorithm over a different one. In this case, it is often best for the lecturer to avoid giving these reasons in the lecture. Instead, they should ask for feedback from the learners in an effort increase the activity in an otherwise passive event. Adding this activity encourages active listening, which, in turn, facilitates observation in learning.

Internships & Apprenticeships

The apprenticeship approach of years past relied heavily on the concept of using observation in learning. Often, the apprentice was merely required to be around the skilled craftsman while the craftsman did his or her work. Of course, the apprentice was required to do many menial tasks during that time. However, the exposure to the successful use of the tools, the process, the problems and the potential solution strategies provided the learner with a chance to integrate that information into their own knowledge base. As the level of comfort grows, the apprentice begins to attempt hands-on work and learn by trial and error. Failed attempts often led to further demonstrations by the expert as the apprentice watched.

Students may find great value in internships related to their field of study. These arrangements provide the learner with an opportunity to observe that is valuable to their future learning in the classroom. An internship can provide them with a more complete picture of what is necessary to function in the field. At the very least, an internship can clarify purpose and provide motivation for learners as they continue through the CS program.

3. Conclusion

Instructors of Computer Science classes at the post-secondary level can certainly aid learning by promoting the use of both phases of observation in the classroom. Many of the strategies discussed here are likely to be similar or very close to techniques used by many instructors. However, we must consider whether or not we take each strategy as far as it needs to go in order to promote observation for learning. Even highly successful educators discover that their techniques require adjustments and alterations over time. It is hoped that this paper will challenge readers to critically evaluate teaching events in their classroom and alter them to better support the learner.

References:

[1] Brookfield, S.D., The Skillful Teacher: On Technique, Trust, and Responsiveness in the Classroom, Jossey-Bass, Inc., 1990.
[2] Bruffee, K., Collaborative Learning: Higher Education, Interdependence, and the Authority of Knowledge, The Johns Hopkins University Press, 1993.
[3] Bateman, W.L., Open to Question: The Art of Teaching and Learning by Inquiry, Jossey-Bass, San Francisco, CA, 1990.
[4] Tafoya, T., *Coyote's Eyes: Native Cognition Styles*, Journal of American Indian Education, Special Issue, Aug, 1989, pp. 29-41.
[5] Eble, K. The Craft of Teaching: A Guide to Mastering the Professor's Art, 2nd Ed, Jossey-Bass, 1988.
[6] Allan, J., Learning Outcomes in Higher Education, FT Magazine: Studies in Higher Education, 21, (1), Spring, 1997, P. 245+.
[7] Brookfield, S., Becoming a Critically Reflective Teacher, Jossey-Bass, 1995.
[8] McKeachie, W., ed., Learning, Cognition and College Teaching, Jossey-Bass, 1980.