Questions to Enhance Active Learning in Computer Science Instruction

J. Philip East and Mark Fienup
Computer Science Department
University of Northern Iowa
Cedar Falls, IA 50614-0507
east@cs.uni.edu
fienup@cs.uni.edu

Abstract

That teachers should use questioning as a part of teaching is considered a no-brainer. However, our efforts at asking questions in class have often been less than successful. Additionally, how one applies general ideas about questioning to computer science courses is not always immediately obvious. This paper provides some background information about questioning, supplies some process suggestions for those wishing to enhance their use of questions, and provides some computer-science-specific examples of questions.

1. Introduction

Learning is not a spectator sport! Yet, the majority of classroom time is spent lecturing. Students need to be more engaged than listening and note taking allow! Studies have shown that after 10-15 minutes of lecturing students essentially stop learning, but their attention-span clock is reset by interjecting activities to break up the lecture. (Stuart & Rutherford 1978) Additionally, we (as teachers) have all discovered that we learn a lot about a topic when trying to teach it and that students often find their own programming errors as they talk through their programs. Active learning is important. Fienup (2000) explored the use of active learning in computer science and asked, why don't we allow or encourage our students to learn concepts at a deeper level by explaining it to or discussing it with other students?

As instructors, we are continually striving to improve our teaching and our understanding of the learning process. For several years we have realized that traditional lecture is too passive and probably is not the best use of in-class time. While traditional lecture might be useful for disseminating information, textbooks and web pages already do that. Why spend valuable class time telling students what the book says. Research studies have shown what we all suspect, lecture is not very effective. Students retain only a small fraction of the material covered, attendance only has a marginal effect on performance, and learning via lecture is independent of the lecturer's quality. (Stuart & Rutherford 1978)

At previous MICSs, we have, jointly and separately, examined new (to us) pedagogical approaches and reported on experiments with our teaching (East, 2001; East, 2000; East, 1999;

East & Fienup, 1990; Fienup, 2000). This paper continues those efforts. The principal reason for producing this paper is too allow us to examine, and hopefully improve, our own teaching. We could accomplish our goal without a paper, however, writing the paper forces us to more carefully analyze our activity. Additionally, we believe that careful examination of teaching and learning is reasonable scholarship when our examination is communicated to others. We hope our efforts can lower the threshold of effort necessary for someone else to try something new. Additionally, we hope to encourage others to more systematically include questioning in instruction and reflect on the results.

We accept as fundamental that it is desirable to have "engaged" students who "actively" process the content we attempt to teach them. Active learning (rather than passive memorization of content) should be the goal of instruction. Achieving active learning is, however, not necessarily easy. In particular we were less satisfied that we anticipated with our initial attempts at providing active learning activities. Our goal became to better understand the art and science of asking questions in class so that our students would more or better by being actively engaged in the content of our courses. This paper is a report of our current understanding of asking questions that enhance active learning in computer science classes.

We both have had some workshops on active learning and have read in this area. However, we have read further in order to refresh our memories and to seek additional ideas. At the same time, we began paying more attention to what we were doing in our classes. In particular, we endeavored to spend some time recording questions we asked and reflecting on the results. We also planned weekly meetings to discuss our reflections and observations. Then we wrote the paper.

This paper represents a work-in-progress. We do not feel we have done enough yet to stop paying relatively close attention to our questioning behavior. Any behavior change that has occurred certainly has not yet become habitual. (Dantonio and Beisenherz (2001) have some nice examples illustrating that early practice is not perfect.) Additionally, we believe there is more to learn.

We discovered that there are a variety of goals that one might have when asking questions. The next part of the paper will discuss various goals for questions and other insights we gained from the literature and our conversations. The bulk of the paper will include exemplar questions and attendant goals. We hope they will be useful to readers who wish to include more questioning in their teaching (and allow some to skip the step where you say "duh" and hit yourself on the forehead for not realizing that there is more to questioning for active learning than just blithely asking questions).

2. Background RE Questioning

We used several techniques for gathering information about questioning. We reflected on our prior experiences with questioning, talked about our experience and our thoughts, and examined readily available literature. From these activities, we identified several goals of questioning in the Computer Science classroom:

- to have students practice a skill
- to grade student performance
- to provide students with practice on applying knowledge
- to motivate a topic
- to motivate students
- to gauge student understanding
- to engage students in active learning
- to develop students' meta-knowledge
- to regain/reset student attention spans

In examining the literature (e.g., Dantonio & Beisenherz, 2001, Chuska, 1995, Wasserman, 1992; Wilen, 1991), we encountered similar lists. For example, Wilen (1991) indicates that

> Although the two major enduring purposes of teacher questions are to determine student understanding of basic facts associated with specific content and to have students apply facts using critical thinking skills, educators have suggested other related purposes:
>
> - to stimulate student participation
> - to conduct a review of materials previously read or studied
> - to stimulate discussion of a topic, issue, or problem
> - to involve students in creative thinking
> - to diagnose students abilities
> - to assess student progress
> - to determine the extent to which student objectives have been achieved
> - to arouse student interest
> - to control student behavior
> - to personalize subject matter to support student contributions in class (p. 8-9)

Both these lists can probably be condensed. They do, however, suggest rather strongly that a variety of goals may be achieved via questioning and that the questioning activity is not simple. Additionally, we also note that the results of questioning activity can probably be classified as recall of knowledge and application of knowledge (understanding).

From our perspective, recall of knowledge is important but probably does not constitute active learning (which is our goal). We might, however, legitimately use a recall question to achieve a goal such as grading a student, i.e., ask a question, wait, call on a student, and record the student's success or failure. Presumably, when the question is asked, all students will engage their minds on the content of the question and formulate a response. This engagement, however, seldom has much depth and probably is not particularly useful instructionally.

While we seldom use such questioning to grade students, we do find ourselves asking these kinds of questions on occasion. Presumably, our goal is to get the students' attention by asking something rather than continually lecturing. Breaking up the lecture is desirable, we just need to be a little more thoughtful as to how we do so.

Another goal of questioning that might involve questions of recall is to assess student knowledge and understanding. Clearly we cannot know what resides in students' heads. To be effective in our teaching, however, requires that we attempt to determine and change (where necessary) what is in students' heads. Indeed this is probably the major tenant of constructivism—that students

construct their knowledge from their experience which is unique to each individual. Thus, before we can influence an individual's understanding, we must gain insight into how that person understands the content at hand. So, questioning toward this goal is important. Such questions need to probe for insights into student understanding rather than request recall of stored knowledge. Achieving the goal of gauging student understanding will, we believe, require substantial time for each student. Additionally, each student's understanding will be somewhat unique. It appears to us that there is some risk of "wasting" class time if this kind of questioning is overused. Additionally, a similar result can probably be achieved when other "higher-level" questions are asked. Thus, we suggest that some questioning to gauge student understanding is useful but the use of questions whose sole purpose is to do so should probably be limited.

A third goal of questioning might be to provide motivation. We wish to motivate students to behave in certain ways. Additionally we want to motivate topics, i.e., stimulate interest in or attention to upcoming topics.

In motivating students, we want to encourage students to do (or have done) assigned readings or activities. This goal is very similar to the grading goal mentioned above, but in this case grading is inconsequential insofar as a goal is concerned. The principal goal is that students do the work and the grading is done only as an incentive. Questions of this sort will generally ask students to recall specific bits of knowledge or outcomes from some assigned activity. Quizzes over assigned reading would fall in this category.

Rather than motivating students, we may wish to motivate a topic. By asking questions about the topic, we might hope to generate student interest in the topic, to expand their conception of the bounds or content of the topic, or to show the utility of some language feature or programming practice. We need further thought and work before making suggestions about the first two aspects of motivating a topic. The third case, however, is quite straightforward. For example, showing repeated code is a motivation for a looping mechanism.

The goal in which we are most interested is that of engaging students' minds on a particular topic in a relatively restricted way. We see the role of such questions to be one of initiating intellectual activity in student minds. Such activity might involve:

- practice of some specific intellectual activity, e.g., designing, testing, debugging, interpreting specifications, etc.
- applying specific knowledge
- having students examine their own knowledge and understanding

While we have approached this goal from the point of view of questioning, we assume we are not restricted to oral questions or even to questions. Asking students to engage in an intellectual activity can be construed as asking a question.

3. What We Think We Learned

3.1 Process Suggestions

Obviously, we suggest that questioning (and other activity) be used to engage students more actively in the content of computer science. But that is not as simple as asking questions. It must be planned. The planning may need to involve a variety of issues and occur at various times and levels in a course.

Before the course begins, we recommend familiarizing yourself with the various goals and types of questions that can be asked and considering the impact on course planning. For example, we believe that there are benefits to having small groups working together on questions. Group formation can be left to students or dictated by the instructor. If the "better" students are spread throughout the groups, there is potentially a teacher per group. Weaker students are more likely to ask questions of their peers. Because students' mental contexts have more in common with students than the professor, the student "teacher" in the group may be in a better position to communicate effectively. We believe that the better students also benefit by trying to explain concepts to weaker students. You should also consider addressing your goals for the class activity in your syllabus and, occasionally, in class. If students understand why you are asking so many questions and otherwise not 'telling" them what they are supposed to know, they may well participate more fully and learn more. You may also wish to incorporate some aspect of grading (e.g., class participation) to reflect your opinion of the importance of active learning.

Before each class or unit, plan your questions. Questions should be used to enhance the learning of the most important topics of each class. Identify the most important content goals or ideas in the lesson. Then proceed to planning your lesson (and the questioning you will use in it). It is as important to consider what you are going to ask as it is to consider what you are going to tell. Do not treat your questions lightly. Consider the goal(s) you wish to achieve with each question. Think carefully about how students will respond to the question.

- Are they likely to just turn off and wait until the "real" classwork starts back up? If so, can you ask the question differently or do something in class that short-circuits that reaction?
- How much time is necessary for them to formulate a reasonable response?
- Is the question clear and unambiguous?
- Is the question too easy or difficult?
- Will students be adequately prepared when the question is asked?

Additionally, consider using non-oral questions. Placing questions on a transparency or handout will demonstrate that you consider them important. Doing so may also communicate to students that you expect them to spend some time on the question while at the same time encouraging you to wait until students have had time to process it. Many students have commented that revisiting questions asked in class an effective way to prepare for examinations since they focus on the important skills and concepts of the course.

What you do during class can affect the success of your plans. When you ask questions, allow students a chance to respond. If students don't respond, wait. If students still don't respond, wait! Eventually, they will respond (if not in today's class, then in tomorrow's). Also, after a student response, wait and think. We find that our first impulse is often less useful than we would have liked. Consider what the student might have been thinking and whether and how you might follow up on the response to enhance the learning of both that individual and other students. If nothing else, when you pause, the students will think you are taking the response seriously.

Be careful how you respond to student answers. You want to foster an atmosphere where students do not feel threatened by answering the questions. Even comments like "that's not quite on the mark, Bob" can be enough to make students hesitant to respond to questions. Since we tend to have groups answering a question, we might simply ask what another group thought. However, it is important that the correct answer is identified as such.

Finally, it is important to spend time after class reflecting on what happened. (Schon, 1983) We often find this hard to do. But, it is necessary, we believe, in order to achieve success at changing our teaching behavior. The up-front planning is quite important but will be mostly wasted if we do not take time to analyze how well the plans worked. In essence, the reflection assesses how well reality matched the plans and, if so, whether the desired outcomes were achieved. Did we actually follow our plans? If not, is that good or bad? Did the students behave or respond as anticipated? Does the planned questioning appear to achieve the desired results? If not, what other questioning or activity might be better? The goal of the reflection is to make us aware of what we do. We suggest a brief reflection time, perhaps keeping a journal or annotating the lesson plan. Of course this data will need to be fed back into the planning process of the next iteration of the course and indirectly for future lessons in the current and other courses.

3.2 Sample Questions

In the discussion below, we provide some examples of questions or class activities. Along with the examples we provide some discussion of our intended goals and of the processes we experienced or expected with the questions. We do not limit ourselves to positive examples. It seems useful to supply some examples of not so good questions so that others might learn from our mistakes.

3.2.1 Knowledge Recall Questions

Knowledge recall questions are relatively easy to ask. Often, however, they do little to enhance instruction. The following questions are probably not particularly helpful, even though they exactly address what we want to know.

- What did you learn in this chapter?
- What are the main points in the reading?
- Do you have questions over the chapter/section?

A small set of quick-check kinds of questions, however, might be useful. They could provide examples of some types of test questions as well as a review of important points in the content. For example, from COBOL:

- For the MOVE statement, when moving a numeric value from a smaller field to a larger field, the result [multiple choice responses omitted]
- What is a file buffer?

From Algorithms

- Some problems are not "feasible". What does that mean?
- What techniques are commonly used to represent graphs in computer programs?

Even though these questions do have some utility, we are inclined to believe they should probably be subsumed into the next category of question in which skills are practiced.

3.2.2 Skill Demonstration Questions

Many relatively simple programming skills such as tracing nested loops are often just demonstrated by professors with the assumption that students have mastered the skill since they did not ask any questions about it. Worse yet, students might fool themselves into thinking they have mastered the skill too. Life would be much easier if we could learn to swim by watching someone swim. Demonstrations of even the simplest skills by the professor should be followed-up by practice questions for the students. The development of skill requires practice, and feedback as to the correctness of practice. Some examples here are:

- Trace the program containing nested loops showing the resulting output.
- What does the given code "do"?
  Similar in nature to tracing this question requires students to abstract from code to a general statement of code purpose. Tracing is necessary for understanding a program and, we believe, skill at abstraction is necessary for coding skill to progress to design skill.
- What does the given error message suggest to you?
- Given the supplied BNF definitions, indicate whether the provided statements/objects are legal and if not, why not? (or provide a legal statement).
- Is the output of this program correct? How do you know?
- Describe a data structure/organization to represent …
- Identify the tasks involved in the given problem. Which task(s) will be repeated? Which must happen before the repetition? After the repetition? During the repetition?
- Identify and correct the syntax errors in the given code?

Other courses have similar examples of relatively low-level skills necessary for competence in the subject—various proof techniques in discrete structures, number representations in computer organization, and counting statements in algorithms.

3.2.3 Questions Drawing on Personal Experience

Questions asking students to draw on their past experiences can often be used instead of asking a more direct, but too complex or abstract, question. For example in Computer Architecture, when discussing immediate-addressing modes with respect to instruction-set-design issues, you might be tempted to ask the question: "How many bits should be used for an immediate operand?" It is more constructive to make the question more concrete by asking students to draw on past experiences by asking questions like the following:

- From your programming experience, what range of integer values would cover 90% of the constant integer values used in all the programs you have ever written?
- How many binary bits would you need to represent this range of values?

The sequence of questions focuses the discussion on the sought after answer.

Questions requiring students to examine their own knowledge and understanding can often be used to motivate a deeper understanding of a topic, but the instructor must be careful that the intended point is made by the activity. To motivate hardware support for operating systems in a Computer Architecture course, I often ask the following sequence of questions:

- What is an operating system (hardware/software, goals, functionality)?
- How does OS/hardware protect against a user program that is stuck in an infinite loop?

The first question motivates the students to think about operating systems and their role. They usually decide that an operating system is software used to provide services such as security, file access, printer access, etc. Students typically answer that the system allows users to break/interrupt a program after a while. Having good oral questions to follow up on student answers is important. Asking about "what happens in a batch system?" steers the discussion back toward the desired answer of a "CPU timer". Other times students respond to the second question with answers like "the operating system will be watching for infinite loops." The instructor might follow up with a question like, "In a single CPU system, how many programs can be executing at once?" If the students answers "one", then you might ask, "If the user program with the infinite loop is running, then how can the operating system (which we decided was a program) be running too?" This gets the discussion back to the need for the CPU-timer hardware support.

3.2.4 Questions to Motivate a Topic

Programming courses have multiple opportunities for motivating topics in that more advanced language features provide capabilities available but difficult without them. Some examples (not in question form) are:

- Repeating the same statements several times for different data can motivate loops, the use of modules, and arrays.
- Carefully wording data structures and algorithms can motivate recursion.
- Extensively nested if statements can motivate a case statement.
- A complex sequential-file-update algorithm can motivate direct access file structures.
- Having students trace a program with no spacing, line-breaks, or comments and with one-character data names might motivate better programming style.

Other contexts for topic motivating questions also exist, e.g.,

- Complete the following statement, "Programming is like _____ because _____". The goal of this question was to get students to reflect on what they knew about programming, hopefully recognizing that it is one of a class of design activities. The question produced some wonderful responses from students. Unfortunately, the instructor did not adequately follow his planned activity and the initial boost from the activity was lost. Next time, the planning will be better (perhaps having students analyze the common characteristics of the responses and discarding some in the process). A lesson learned is that following your plan for follow-up is as important as the planning.

Our COBOL course has a programming prerequisite. The course was introduced with a set of questions that essentially identified the goals of the course, i.e., at the end of the course you should be able to answer the following questions (list is abbreviated here):

- What is the general organization of COBOL programs?
- What is file buffering and how does it work?
- What data "structures" (e.g., records, arrays) are available?
- Does COBOL have modules (procedures/functions)? Local variables and parameters?
- How does the standard looping mechanism work?
- How is data stored in COBOL (bit-wise and structure-wise)?
- What are the typical COBOL tasks and how are they solved?
- What are some typical COBOL applications and some standard solution strategies?

This kind of question is an advance organizer and should serve to establish cognitive hooks into students' past experience. The list of questions should probably be revisited periodically, perhaps

at the end of each chapter to help students recognize their progress and keep in mind that the course has high-level goals in addition to acquisition of knowledge about a particular programming language.

3.2.5 Questions to Create Cognitive Dissonance

A colleague once told me that students in his crystallography course did not have preconceptions about the content in his course. He was wrong. Students may come to us with little knowledge and incorrect assumptions about word usage and meaning, but they will always have some preconceptions about our content. Often the preconceptions will be inaccurate and hard to replace. Identifying and attempting to fix them and to short-circuit the establishment of new misconceptions are critical aspects of teaching. The strongest learning occurs when we are able to produce cognitive dissonance in student minds. We need this kind of learning to alter misconceptions—weaker techniques will not work. Additionally, it would be nice if we were able to generate such a mindset at will. Probably we cannot, but we can try.

An example from a systems course is the following pair of questions.

- How does the OS/hardware prevent a user program from accessing files of other user?
- How does the OS/hardware prevent a user program from accessing (RAM) memory of other user programs or the OS?

The following trio of activities relates to run-time requirements in an algorithms course:

- Sort 1,000 items in an array using the bubble sort.
- Sort the same 1,000 items in a direct-access file using the bubble sort.
- Sort the same 1,000 file items using the quick sort or shell sort.

Clearly, our lists of questions are incomplete. Space concerns make that necessary. So too does our level of progress. Frankly, we have only begun the work necessary to become better questioners (and, thus, better teachers). As we develop our skill, we will share more of what we learn.

4. Conclusions

Our most significant insight is that asking good questions takes work. We had to (and may still need to) read about questioning and apply what we read to computer science. Additionally, relatively significant planning is necessary. In essence, we need to plan for questions, much as we plan for lecture.

We are still convinced that doing the extra work will pay off. We think student learning will improve—more students will learn more of the material at a level we think is good. Additionally, we believe the "extra" work in planning will lessen, and perhaps disappear. As we learn more and practice questioning (and planning for it), the time requirements will be less. Also, as questioning becomes a bigger part of our teaching, the planning of telling is replaced by planning for questioning.

Should you decide to include more questioning in your teaching, we have some advice beyond that of reading and planning. Reflect on your questioning behavior. Explicate your goals and plans before teaching. After teaching, reflect on how well you implemented your plans and on how well the questioning worked. Then introduce those conclusions into your future planning. (This may require some record keeping.) Finally, do not expect perfection. Like all other human endeavors, you will get better with practice, particularly with good (reflective) practice.

5. References

Chuska, K. R. (1995) Improving classroom questions. Bloomington, IN: Phi Delta Kappa.

Dantonio, M & Beisenherz, P.C. (2001) Learning to question, Questioning to learn. Boston: Allyn and Bacon.

East, J. P. (2001) Experience with In-person Grading. Proceedings of the 34nd Midwest Instruction and Computing Symposium, April 5-7, 2001. Cedar Falls, IA.

East, J. P. (2000) Teaching, Assigning, and Grading: Continuing Efforts to Enhance Learning. Proceedings of the 33nd Midwest Instruction and Computing Symposium (formerly the Small College Computing Symposium, on CD-ROM), April 13-15, 2000. St. Paul, MN.

East, J. P. (1999) Providing feedback to CS students: Some alternatives. Proceedings of the 32nd Small College Computing Symposium (on CD-ROM), April 15-17, 1999. LaCrosse, WI.

East, J. P. & Fienup, M. (1990) In search of a model for teaching programming. Proceedings of the 23rd Small College Computing Symposium (p. 236-242). River Falls, Wisconsin.

Felder, R. & Brent, R. (1996). Navigating the bumpy road to student-centered instruction. College Teaching, 44, 43-47.

Fienup, M. (2000) Active and group learning in the Computer Science classroom. Proceedings of the 33nd Midwest Instruction and Computing Symposium (formerly the Small College Computing Symposium, on CD-ROM), April 13-15, 2000. St. Paul, MN.

Frederick, P. (1986). The lively lecture - 8 variations. College Teaching, 34, 43-50.

McConnell, J. (1996). Active learning and its use in Computer Science. SIGCSE Bulletin, 28, 52-54.

Schon, D. A. (1983). The reflective practitioner: How professional think in action. New York: Basic Books.

Silberman, M. (1996). Active learning: 101 strategies to teach any subject. Boston: Allyn & Bacon.

Stuart, J. & Rutherford, R. J. (1978, September 2). Medical student concentration during lectures. The Lancet, 514-516.

Wasserman, S. (1993) Asking the right question: The essence of Teaching. Bloomington, IN: Phi Delta Kappa.

Wilen, W. W. (1991) Questioning Skills, for Teachers. Washington, D.C.: National Education Association.