

Tutorials on Computer Science
Andrea Danyluk, Professor of Computer Science
and Chair of the Computer Science Department
Williams College

I will follow up on Sarah's discussion by sharing with you my tutorial experience – and, more generally, the experience of faculty in Computer Science. I will try to be brief, as I would then like to turn to and focus on tradeoffs in teaching courses as tutorial, rather than as lecture or seminar.

We computer scientists obviously have the same goals as our colleagues. We hope that our students will gain understanding of new topics and that they will develop skills such as creative problem solving, critical reading, writing, and speaking. The tasks we see as supporting these goals and that we value as a department include:

- reading selections from textbooks and primary literature,
- solving problems,
- writing proofs,
- giving presentations,
- writing critically on research in the primary literature, and more generally
- responding to the work of others, including their peers.

These should sound familiar by this point.

In addition, algorithm design and programming play central roles in our courses.

This list of tasks is obviously long. Therefore, one way to describe the range of computer science tutorials is to describe the elements on which we individually decide to focus – or on those we're willing to give up. For example, one of my colleagues, in his tutorial on modern computer architecture, is willing to give up weekly problem sets in exchange for a scheduled weekly lab in which students learn to design chips starting at the transistor level. Another colleague doesn't see programming as fitting into the tutorial model. He prefers to focus on problem solving. I find it difficult to give up anything.

We all have students meet weekly in pairs. In my tutorial, Machine Learning, from which you can find a variety of documents in your packets, I try to match students who are similar along several dimensions, including major, experience, grades, and temperament. I select the weekly readings from the primary literature and a variety of textbooks. In any given week, students can expect to do, on average, two or three different types of exercises in addition to the reading: perhaps problem solving and writing a critical paper on a research article; or writing a program and proving theorems about its behavior.

Every student presents something every week. In fact, students don't know ahead of time who will present what – each student must be fully prepared to participate in the tutorial session in every way. I do incorporate the presenter-critic notion into the tutorial: the student not presenting a given problem must serve as critic; one student must lead off the discussion of a research paper and thus sets at least the initial agenda for the discussion;

in programming weeks, one student presents his or her program while the other serves as critic. At the end of each tutorial meeting, I collect the written work and grade it.

Now that I've given you at least a brief glimpse into computer science tutorials, I'd like to move on to the focus of my remarks – the tradeoffs of teaching courses (and, in particular, science courses) in a tutorial format as opposed to a lecture or seminar format. I will consider the question of tradeoffs in two broad categories: curricular and staffing. On the curricular side, I will try to address what we can do in tutorials that we can't do otherwise (and vice versa). On the staffing side, I will consider the question of how many faculty we can afford to spend on tutorials. I will also consider the issue of fair workload.

Sarah and I interviewed faculty in the sciences and everyone we interviewed mentioned a depth-breadth tradeoff. That is, we all agree that we cover fewer topics in tutorial than we do in lecture. On the other hand, we all believe that our tutorial students gain a much deeper understanding of the material.

In addition to learning the material more deeply, tutorial students develop a variety of skills that they might not in lecture:

1. They learn how to read. When they enter college, students are not necessarily good at knowing on what they should focus when reading. This is especially true for students reading primary literature in the sciences. (To be fair, students can also develop this skill in seminar, and I will address those a bit later.)
2. Students learn to be critical of written sources. One faculty member in my department went so far as to say that he wanted students to learn to be skeptical of everything they read.
3. Students gain confidence in going out on a limb. This relates to the previous point about learning to be critical. It also refers to students gaining comfort in putting forth their own ideas and solutions to problems.
4. Students learn how to express themselves clearly.
5. They learn how to write about science. Many of our students are wonderful writers, but relatively few have had the experience of writing about *science*.
6. Finally, students begin to see themselves as *scientists*. This develops in many different ways, but let me relate one example that I heard in my interviews. In tutorial students will push on the material in all sorts of ways, and it isn't uncommon for a student to raise a question to which the instructor can only answer, "I don't know." When this happens, though, the instructor and students can work together to find the answer to the question, either by deriving it or by thinking about where they can find it. In this way, by watching and emulating the instructor and by working with the instructor, the students begin to learn *how to do science*.

There is, of course, a down side to tutorials. As I noted earlier, the faculty we interviewed agreed that we cover fewer topics in tutorial than we do in lecture.

Another dilemma for the sciences has to do with role of problem solving in our disciplines, as Sarah discussed. Pedagogically, we believe that it is critical that all students work the problems every week. On the other hand, we want to maintain the presenter-critic paradigm of the tutorial and to fairly balance student workload.

Returning now to the benefits of tutorials, faculty can better tailor tutorial courses to the needs and interests of their students. (As others have discussed,) faculty work hard to “engineer” good tutorial pairs. The separation of students into pairs is a mixed bag, however. On one hand, as one faculty member put it, “everyone runs their best.” Students can strive to do their best without feeling pressure that grows from their own perceptions of their classmates’ progress. Ironically, the tutorial might even best serve those students we sometimes fear won’t take them – that is, the weakest students. The tutorial provides a natural way for an instructor to spend focused time with those who might otherwise lag behind.

On the other hand, students lose the benefits of having a point of comparison. In the absence of peer reinforcement of expectations, some students might believe they’re doing well, when, in fact, their classmates are pushing on the material in more advanced ways; others might believe they’re the only ones struggling, when the class as a whole might be having difficulty. A faculty member in Biology typically rearranges tutorial partners mid-semester, which helps her students to recalibrate to some extent.

Some faculty are concerned that, in addition to potentially losing the ability to calibrate, students in a tutorial lose the sense of community that might be more readily established in a seminar or in a lecture. A faculty member in Chemistry said that with six or fewer students enrolled in a course, he leans toward the tutorial format; with larger numbers, he leans toward a seminar or lecture, because the larger numbers make the group discussion better. Professors in Biology cited this as one of their reasons for teaching upper level reading courses as seminars, rather than as tutorials.

One member of my department, as I mentioned earlier, holds weekly scheduled labs in conjunction with his tutorial. The pedagogical reason behind the labs is to give students a more practical perspective that complements the theoretical perspective of tutorial sessions. My colleague also believes that the labs are important because they build community. In fact, for the final lab assignment, which is an open-ended student-choice project spanning three-four weeks, he gives the students the option of working alone or in small groups or as a class. Every time he’s taught this tutorial, the students have elected to work *as a class*.

A faculty member in Geosciences, on the other hand, feels that labs hurt the tutorial session dynamic. He believes that, as soon as you introduce a more traditional course component, where the professor is in control, the students tend to feel less individual responsibility for the material and the tutorial session dynamic changes.

This discussion of labs is a convenient point to turn to the second type of tradeoff I mentioned earlier – that having to do with staffing and workload.

Lab work is obviously of central importance in the sciences and is a vital component of many of our courses. Many of us believe that, if done in the right spirit, a tutorial with a laboratory still fits the fundamental definition of a tutorial. This is not the issue. The issue, which is related to workload, has to do with the fact that in certain advanced courses, lab work involves the use of *expensive* instruments that must be shared by the students. In such cases, rather than running a lab with 18 students at a time, which most of our departments can do at the introductory level, some departments must run upper level labs with, say, 4 students at a time. This increases the number of meetings/week for the instructor. Assuming an enrollment of 10, which is the enrollment cap on tutorial courses at Williams, the instructor has to run 5 1-hour tutorial sessions and 3 lab sessions each week. An equivalent lecture course would typically require 3 50-minute lectures and 3 lab sessions.

The Williams cap of 10 students per tutorial that I just mentioned is in place to help faculty teaching tutorials to maintain a reasonable workload. Even if we ignore the lab issue, the cap presents a problem for courses that typically get enrollments that are significantly higher. In those cases, should students be dropped? If so, where do they go? What sort of burden does this place on the enrollment levels of courses that are not taught as tutorials?

If students are not dropped, is the burden on the tutorial instructor unreasonable? If the instructor is compensated somehow, what are the implications for other faculty? For the department's curriculum? For example, if a tutorial with 20 students counts as two courses, what offering must the department drop in exchange for the additional section of the tutorial?

Rather than end my discussion of tradeoffs with workload issues, let me return to one final curricular benefit. Earlier I itemized the many benefits of tutorials for *students*. However, a benefit cited by several faculty was the extent to which *they* learned more through tutorials. As one said, because students in a tutorial have so much control over the direction of a discussion, they can push you to confront the gaps in your own understanding. Another faculty member even said that by discussing papers several times, he gets new research ideas.

As one of my colleagues in the Computer Science Department noted, in our department we've never twisted anyone's arm to teach a tutorial. Yet we see our faculty fighting (not literally) for the opportunity to do so. Our difficulty is not in finding courses to teach as tutorials or faculty willing to teach them – it's the need to cover our lower level and core courses, which we don't see as amenable to the tutorial format. We've also felt it important to offer a non-tutorial elective option each semester, in part to handle high enrollments and in part to prevent a situation in which a student is forced to take a tutorial. Because a student's experience in our tutorials depends so much on the attitude of his/her partner, we see nothing worse than having a student in a tutorial who simply doesn't want to be there.

On the other hand, an informal survey I recently did indicates that perhaps we don't need to be so worried about this. I reviewed the transcripts of Computer Science majors in the classes of 2006 and 2007. I counted the tutorials taken by each of these students for the Computer Science major. (So if a student had taken a tutorial in, say, Physics or English, I didn't count it. I did, however, count tutorial versions of Mathematics courses that are required for our major. It's important to note that non-tutorial versions of the Mathematics courses are available every semester.) In the class of 2006, only 2 of 9 Computer Science majors graduated without taking a tutorial for the major. This year, only 3 of 16 majors will graduate without a tutorial for the Computer Science major. Of those who have taken tutorials, some will have taken 3, 4, and even 5 tutorials for the major. In our current junior class, the class of 2008, 6 of 13 majors have already taken tutorials and half of those have taken more than 1.

Sarah's survey of Physics majors gives even better numbers – better in the sense of a higher percentage of students graduating with tutorial experience. Admittedly, it is somewhat more difficult to avoid tutorials in Physics than it is in Computer Science, especially for grad school bound students. But it isn't impossible. Last spring in senior exit interviews that Sarah conducted with Physics majors, she asked them directly about tutorials and their place in the Physics major. Specifically, she asked whether they thought it was worth teaching these courses as tutorials, pointing out that if the department didn't teach them as tutorials, they might be able to offer more upper level electives. One non-grad school bound student said he found the skills he'd learned in tutorial to be very useful in job interviews. In general, most said they thought that the tutorials were very worthwhile, in terms of “teaching us to learn material on our own,” “teaching us the huge skill of standing at the board and explaining ourselves,” and last, but not least, “taking us to another level.”