Mixed Nuts: Atypical Classroom Techniques for Computer Science Courses

by Sid Stamm

Introduction

AAAAAAAAAHHHHHHHHHHHHHHHH!

Didn't expect that, did you? Neither did Steve Wolfman's class when he let loose a blood-curdling scream on the first day. His goal was to demonstrate the strength of community and pooled resources. This point was clearly illustrated when he asked the entire lecture hall of two hundred students to scream at the top of their lungs. Why? He was using an atypical teaching technique to get the attention of the students and keep them interested. Students do not always learn all that the instructor is trying to get across a lecture. In fact, it is likely that even students who can get the most out of a lecture grow restless and might be more involved in class if it were taught with a twist. In order to capture the interest of students more effectively, instructors could use atypical classroom techniques. Moreover, unlike lecturing and giving homework, these unorthodox techniques can also keep students attentive and target preferred learning styles. This article presents some experimental and anecdotal evidence to support the theory that the use of these techniques improves students' learning in an introductory Computer Science (CS) class.

The Problem

Students do not always pay attention, especially during long lectures. They lose interest, because they do not feel like they are learning, or sometimes they do not care about what is being taught. Being an effective teacher is not always a simple task.

Interest

In many cases, an introductory CS class is a general requirement, therefore, a diverse student body is enrolled; not just CS majors. A common complaint among students is that the material is not interesting. Many people studying pedagogical techniques agree that "students who are new to computer science typically find the field full of theoretical, technical, or even tedious concepts" [6]. The lack of interest is likely attributed to the large number of building blocks, such as procedures, syntax, and algorithms that are initially taught to students so that they can apply these to more complex and interesting problems. Since the students are exposed to this type of information at the beginning of their studies, students who have no initial motivation to learn the basic CS concepts lose interest in the subject early on.

Concepts not Easily Taught

Applicability an incentive for students to learn; the lack of it can make a course seem irrelevant. Still, this is not the only factor that limits their motivation to learn. CS concepts can be difficult, at first, because they are typically alien. Students often need lots of time to gain enough familiarity with the material to understand and apply a concept. In some disciplines, this can be solved by requiring the students to perform laboratory experiments. However, experimentation is difficult for entry-level CS students. The students must know a programming language to experiment with concepts taught in the classroom, which can add to the frustration of learning things that cannot be immediately tested.

Short Attention Span

In order to adequately expose students to CS concepts, they need to get their hands in the proverbial Computer Science dirt: they must learn both the theory and the syntax to apply the ideas. The instructor must put in more effort to involve students who are not initially interested in the class to keep their attention. Steve Wolfman, of the University of Washington (UW), says "there is a need to jar people at the beginning of each class to shift them from what they have been doing to what we are going to be doing" [5].

In 2002, Wolfman was given the Excellence of Teaching Award by UW. The UW Computer Science and Engineering Department Chair, David Notkin, says that "Steve has outperformed—by a substantial amount—every faculty member who has taught the other 114 recorded offerings of these courses" [5]. Wolfman has shown that using atypical techniques to shift the students' attention is effective and that such a shift is especially important for instructors who teach first year college students. Usually, these students take classes in many different subjects in a given semester and may have difficulty switching their modes of thinking. For example, an instructor's effectiveness decreases if some of the students are worrying about a physics exam that took place earlier in the day. Gaining their attention takes up a significant percentage of the course and leaves the professor less time to actually teach. Recently, the computer science department at Rose-Hulman has moved the entry-level CS course from four 50-minute blocks to two 100-minute blocks, because it took students too long to get settled into the class. This has since proven beneficial because the students are spending a smaller percentage of the class period setting up and adjusting to the new subject matter, thus giving instructors more time to teach.

Various Learning Styles

Bonwell and Eison claim that "some cognitive research has shown that a significant number of individuals have learning styles best served by

pedagogical techniques other than lecturing" [1]. Thus, the lecture/homework approach may not be the optimal teaching method to use in any discipline because instructors can lose the attention of the class during a lecture. Getting students to scream, as Wolfman did, is often a good way for an instructor to keep their attention, at least for a little while longer. However, every student learns differently, so the surprise screams or other techniques can come from completely different parts of a classroom education. Therefore, a good way to keep the attention of most students in any class would be to use techniques that cater to different learning styles. Richard Felder of North Carolina State University describes four different learning style models and classifications of students. He writes that "if professors teach exclusively in a manner that favors their students' less preferred learning style modes, the students' discomfort level may be great enough to interfere with their learning." In any given class, a professor may have students with a variety of preferred learning styles. Felder suggests that instructors do what he calls "teaching around the cycle" [4]. In his paper, he describes these different learning styles, and methods of teaching that benefit each style.

The Solution

Based on this research, I believe that concepts should be taught with as many different styles and methods as possible in order to reach a broader audience. Clearly, the classical lecturing technique is not as effective as it could be. What steps can an instructor take to make the learning experience better? First, the instructor should do his/her best to gain and maintain the attention of the students. Second, the instructor should cater to different learning styles. Third, concepts should be covered more than once and in different ways. Keeping these three goals in mind will help an instructor make an introductory CS class more effective.

Gain and Keep Students' Attention

A good way for instructors to get their students' attention is to surprise them. Students are fascinated with unpredictability, and are more likely to watch an instructor if they can expect the unexpected. A teacher that has students' attention has the opportunity to motivate them to learn; the hard part is keeping it. Because each student might be interested in something different, it can be difficult to hold the attention of the entire class. To address this, an instructor must use various teaching methods to capture as many students as possible. Using different classroom techniques can help an instructor address a variety of students, while at the same time engaging others in the material.

Teach Around the Cycle

Once students are engaged, an instructor must be sure to teach in ways that will help the students learn. Since students have various preferred learning styles, a professor needs to vary the method of instruction. For example, a way to teach to students who prefer active learning can be to relate the material as a metaphor. Getting them involved in a story and later telling them how it relates to the subject can spark the learning surprises that professors frequently desire. Sometimes students prefer something other than typical facts to learn a concept. In this case, teaching by telling a story can be quite effective. For example, an instructor could construct a metaphor to compare it to an implementation. Once the students are involved, they can be asked how they would behave or feel and it can be related to one of the attributes of a desired concrete example. This process is sometimes referred to as "extended analogy" [6]. These various methods can help an instructor teach "around the cycle" or vary the method of instruction so that all of the students have an equal opportunity to learn in their preferred manner. At the same time, catering to multiple learning styles can help an instructor reiterate concepts without seeming redundant.

Reiterate Concepts

One of the best ways to learn introductory CS concepts is to be exposed to them repeatedly. The obvious problem with this is that ideas can become redundant. If an instructor uses many different teaching methods for explanation, then the repetition will be less noticeable because the instructor brings in a new way of looking at each concept. Such variation benefits both the professor and the student. The same daily routine can become just as boring for an instructor as it can for the students. Atypical techniques can provide relief from such a monotonous routine. They can also provide a way for the students to help direct the class. The instructor can find out how to teach them more effectively by seeing how different individuals learn.

Using Atypical Techniques

An atypical technique is something unusual or different from the common style of classroom instruction, which consists of lecture, homework, group work, and tests. Example techniques include role-playing exercises, giving bonus points to students for subject-related jokes, in-class debates about the outcome of a situation, or an extended analogy [6]. There are countless variations of atypical techniques.

Active Learning

Techniques in this category heavily rely on student involvement and often require the students to get up and move around. These put the students "into" the CS concept being taught. These techniques primarily target the students who learn best by involvement rather than those who learn best from watching or listening and may include:

- **Setting up a debate**: Break the class into groups or select a few students to take opposing sides of an issue or different solutions to a problem. Have them argue for the accuracy or use of their side [1].
- **Role-playing**: Put students in an activity where they act as parts of a computer program, or have similar behaviors to some aspects of a CS topic. They can learn why things behave the way they do or discover the possible benefits for a type of implementation or system [1].
- **Simulating algorithms using the class**: Use the class as data members in an algorithm or process. Distribute papers based on a set algorithm or have the students alphabetize themselves [7].

Lecture Modifications

These techniques are modifications on the delivery or protocol for a lecture. They have some form of shock or entertainment value and are primarily used to get or keep students interested and may include:

- **Jarring people at the beginning of each class**: Steve Wolfman makes a point to do something unpredictable and usually surprising at the beginning of each class, such as screaming [5].
- **Using an extended analogy**: Make up a story that technically has nothing to do with any CS concept, but somehow relate it, allegorically or metaphorically, to CS. Camp, Hooper, and Matocha, write that "an item, exaggerated to the point of silliness, becomes easier to recall" [6].

Some Trials

The best way to test the effectiveness of atypical techniques is to expose a class of undergraduate freshmen non-CS majors and examine the results. Two experiments were conducted to test the theory that atypical classroom techniques would aid the students in learning. For the experiment, the technique of role-playing was the primary concentration. The students that have trouble learning from a lecture or from programming in a language as complex as Java, this should provide an alternate method to understand most CS concepts.

Setup

Two independent sections of the entry-level CS course (CS120) at Rose-Hulman were used in the experiment. The two sections met at different times and were equal in size at initial enrollment with thirty students each. Almost all of the students had a declared undergraduate major outside of computer science; out of the 60 students, only one had declared CS. The instructors for each section of CS120 (Dr. Cary Laxer and Dr. Andrew Kinley) had taught entry-level courses in the field and each had their own preferred teaching styles. Since the section that Laxer teaches met earlier than Kinley's, Laxer's class was used as a control variable. Using this setup, students could talk between sections about how the concept was taught and it would not affect the experiment. Laxer's class is referred to as section C, for control, and Kinley's class as section E, for experimental.

Initial Bias

In a small college environment, like Rose-Hulman, most classes and professors have a reputation of some sort. Any bias of students toward the class and professors in the experiment were gauged. If the two sections had different expectations for the class or professors, it would be taken into consideration when reviewing the experimental results. On the first day of class for each section a simple, anonymous questionnaire, asking questions of the students about what they had heard and what they expected from the professor and the course, was passed out. According to the anonymous questionnaire, no student had taken a course from the named instructors.

Section	Control	Experimental
Course Relevance	4.23	4.17
Subject Interest	3.83	3.66
Instructor Opinion	2	11

Table 1: Initial Student Bias.

For the course relevance and subject interest, students were asked to rank how relevant they think the course is and how interested they are in computer programming, each on a scale of one to five (one being lowest and five being most interest or relevance). The numbers (Table 1) are the arithmetic mean of the result set. The results show that students in the control section are slightly more interested in and see the applicability of the course more than the experimental section. However, on the whole the students are only slightly interested in computer programming. The students were also asked to state what they have heard about the professor who taught their section. Negative responses were given a negative point (–1), a neutral or no response a zero, and a positive response a positive point (+1). The sum of the students' opinions (Table 1) is a positive number, which indicates an overall good opinion. On average, the students in the experimental section had been given a higher opinion of their professor.

Section	Control	Experimental
C++	27%	24%
Java	2%	3%
Any Language	57%	66%

Table 2: Previous Programming Experience.

Finally, the students were asked what languages they knew or had used. Table 2 shows the experience of each section by percentage. Approximately one fourth of the students had been exposed to Java (which is used in the course) or C++, and well over half had been exposed to programming of some variety. The base skill level was close to even in both cases, so programming experience was not expected to influence the results. The instructors for both courses seemed enthusiastic about the experiment and were open to try new techniques in order to help the students learn more. Since they both seemed in favor of trying atypical classroom techniques, their attitude during instruction should not have had a significant effect on the students.

Expected Effect of Bias

Based on the analysis questionnaire, the students were fairly optimistic concerning the course's relevance and the interest in programming was about the same. The students in the experimental section had been given a significantly higher opinion of their instructor, which could make them more receptive to an atypical technique. On the other hand, high expectations for the professor could give the students less motivation to learn on their own. This could lead to a class of spectators that would spend less time learning outside of the class than the other section. Very few of the students knew Java, so this should not have affected the results. Overall, the students in both sections brought a similar amount of programming knowledge and motivation to the class as it started. The experiment is not significantly biased toward motivation and programming skills.

Experiment 1: Interfaces

The first experiment was conducted on the concept of Java interfaces. The control class was taught this concept using the standard lecture and group work. The experimental class was subjected to a simple role-playing exercise that supplemented the standard lecture. During the execution of this experiment, Dr. Kinley taught both sections of the course, because Dr. Laxer was unavailable. Since the experiment was centered on one lecture and the same instructor was teaching both sections, this first experiment only compared teaching styles between sections. The goal for this experiment was to show the effect a role-play illustrating the functionalities of Java interfaces had on students. It was hypothesized that the students would understand interfaces and their uses more easily than a class that was just taught by only lecture and homework.

Execution

On the day that the students were to learn about Java Interfaces, both sections were given an identical quiz with questions pertaining to the

concepts they were to learn. This pre-lecture quiz contained questions that the class should not have known before the class as well as some they might have been able to answer. The quizzes were then marked and the class' overall mean, median, and standard deviation were calculated.

The students were then taught the concept of Java interfaces as defined in the course curriculum. Dr. Kinley conducted a role-playing exercise with the experimental class. The next time the class met, they were given a nearly identical quiz with slightly reorganized questions but with the same information as the pre-lecture quiz. Again, the quizzes were marked and the class' overall mean, median, and standard deviation were calculated.

The Role-Play

One student (call him Jeff) was told to leave the room, and another (call her Jill) was told to come to the front of the room. Jill was given a list of activities that she would "know" how to do if asked. Once she felt comfortable with the list of activities, Jeff was invited back. The instructor told Jeff to give Jill different instructions. After a few moments of Jeff asking Jill to do activities that she could not do, the instructor explained that this is how life is without Java-style interfaces. Once again, Jeff was told to leave the room. This time Jill was given a set of activities she had to be able to do and what kind of parameters went with each instruction, and they were then written on the board. Jeff was invited back in and found himself greeted by a list of things that Jill could do, as well as what she needed to know to do them. The instructor asked if it would be easier to tell Jill what to do this time. As expected, Jeff confirmed it was easier and found that Jill would react to his instructions in the way he expected. The instructor explained that the writing on the board was a kind of Java-style interface.

Initial Instructor Reaction

Immediately after the classes, Kinley explained that the students were very receptive to the role-play exercise. He said their enthusiasm toward class participation increased when they got to stand up and do something rather than just sit and listen during the class. The mood in the classroom was almost cheerful and he felt that he had the attention of the entire class through the exercise. Kinley said that he believed the students in the experimental section learned the concept of Java-style interfaces much more quickly than the control class. He also believed that they were more able to apply their knowledge of interfaces than the class that had not participated in the role-play. He expressed his confidence in the effectiveness of the role-play in getting the students to understand and apply interfaces.

Statistical Results

However, the statistical results are not as straightforward as the professor's anecdotal confirmation of the hypothesis. Percent change (Table 3) was calculated for each student. Data in this chart reflects the mean, median, and standard deviation of the changes, and not the change in mean, median, and standard deviations. In short, the control section seemed to show more improvement than the experiment section. On average, the students in the control section improved their scores 11% while the experimental section showed a much smaller mean improvement of 3%. The experiment section started with half of the scores above 67%, but finished with a median of 60%. Seven percent of the students in that class moved below the average, indicating that either the tail end increased (fewer very low scores) or there were more very high scores than before the lesson. Combining this 7% decrease in median with the fact that half of the students' scores decreased significantly (-10% median in change) leads to the conclusion that many students did worse than average, but fewer students had extremely low scores. This spread in the spectrum is verified by the large deviation in the change of each of the student's score.

Control Section	Pre-Quiz	Post-Quiz	Change
Mean Accuracy	50%	61%	11%
Median Score	60%	60%	5%
Standard Deviation	33%	15%	30%
Experiment Section	**Pre-Quiz**	**Post-Quiz**	**Change**
Mean Accuracy	56%	59%	3%
Median Score	67%	60%	-10%
Standard Deviation	39%	23%	45%

Table 3: Results of the experiment.

The median change for the control section is positive, which suggests that more than half of the students in the class had increased scores. The mean increased quite a bit (11%) as well, suggesting that there are more high scores, but fewer perfect scores.

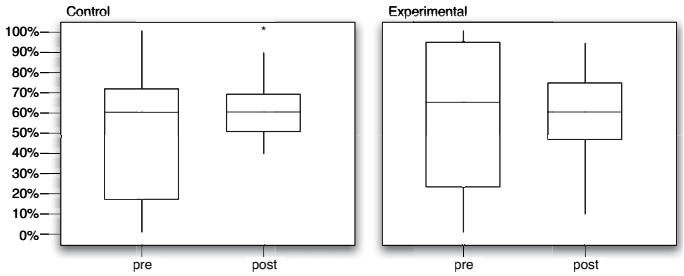

Figure 1: Interfaces Test Results.

The standard deviation of each section indicates that the scores clumped together much more closely around the mean after the lesson. This signifies that more of the students in both cases are ending up with a closer to average mastery of the class; each individual's skill in Java-style interfaces is becoming more "normal." In summary, both sections' scores coalesced into closer-knit groups around their averages (Figure 1), but many more students in the experimental section experienced a decrease in scores. In both cases, there is a variance of scores below the average that shrinks after the lesson. It seems that the people who were having the most trouble in the experimental section were helped by the role-play, but some people in that section suffered as well.

Experiment 2: Threads

The second experiment was conducted on the understanding of threads. The control class was taught this concept using the standard lecture and group work method. The experiment class performed a simple role-playing exercise to supplement the standard classroom routine. During this experiment, Dr. Laxer taught the control section and Dr. Kinley taught the experimental group. This time, with the use of two different instructors, a possible difference in teaching styles had to be

taken into consideration Once again, the goal for this experiment was to show how a role-play illustrating the functionalities and behaviors of threads would increase the students' understandings of the concept.

Execution

Execution was exactly the same as in Experiment 1, but with a different role-playing game to illustrate the concept of threads.

The Role-Play

The second role-play was conducted in the manner of a live *Guess-Who?* [2] game. One person would hunt for a specific person by asking questions and eliminating students based on physical traits. One student (call him Jeff) was chosen to be the leader of the exercise and was told to leave the room while the game was set up.

The instructor then chose one student at random and gave that student a treasure to hide on a person. Once the student had hidden the treasure, the instructor asked the entire class to stand up. Jeff was asked to come back in, and was told he needed to find the student with the treasure by process of elimination. The process was to be iterated until only one student remained:

1. Ask a question about the person's appearance. For example, "Is the treasure held by a student with blond hair?"
2. Go through the class, person by person, and tell each person that did not fit the criteria to sit down and each person that did to remain standing.

Once the treasure was found, Jeff was asked to leave the room once again. The instructor chose three people who would act as Jeff's threads, and they were told to join him outside the room. Again the treasure was hidden on a random student and the entire class was asked to stand up. When the treasure was again hidden, Jeff and his threads were asked to re-enter the room. He was told to do the same thing, except this time he could dispatch an idle "thread" to sort through the class. This added the option to have multiple people working at once. The class could now clearly see the benefits of multitasking and thus, the benefits of having multiple threads in a computer program.

Initial Instructor Reaction

After the role-play class, Kinley explained that the students really enjoyed the exercise. He said that the entire class was energized, awake, and engaged in the learning of threads. The change of pace was refreshing for the students as well as the instructor and everyone was more interested than usual. Once again, the instructor said he believed the experiment was effective because the students seemed to learn a large amount from the exercise. Not only did it interest them and get them involved, but the analogy of dispatching additional aid seemed to help the students understand the purposes, benefits, and functionality of threads.

Statistical Results

Percent change (Table 4) was calculated for each student. The data reflects the mean, median, and standard deviation of the changes, not the change in mean, median, and standard deviations. At first glance it seems that, on average, the experimental section improved more (28% mean increase versus 5%), but at the end of the exercise the control section still had a slightly higher mean score and thus a deeper understanding of threads. In both cases, the standard deviation shrunk to 13%, which indicates that the spread of scores lessened after the class on threads.

Control Section	Pre-Quiz	Post-Quiz	Change
Mean Accuracy	78%	83%	5%
Median Score	83%	83%	3%
Standard Deviation	16%	13%	17%
Experiment Section	**Pre-Quiz**	**Post-Quiz**	**Change**
Mean Accuracy	53%	82%	28%
Median Score	50%	83%	33%
Standard Deviation	23%	13%	28%

Table 4: Results of the experiment.

The experimental class, however, started with a bigger spread than the control class and ultimately coalesced more. This would indicate that more of the students moved closer to the mean. The median score in the control section did not change at all, which implies that the lower scores moved an insignificant amount, and that the scores that started above the median remained the same as before. For the students in the class above the median this session did not affect their knowledge of threads. The median in changes for the control section is very low, which indicates that most of the students had little score change, if any. The median score in the experimental section followed the same upward trend as the mean, which indicates that the increase happened class-wide and not just with a select few students. The fact that the standard deviation dropped by about 10% indicates that the benefit had the biggest impact on people below the average, because their scores grew much closer to the mean. Overall, it seems the experimental section made the most progress over one day with the lecture on threads.

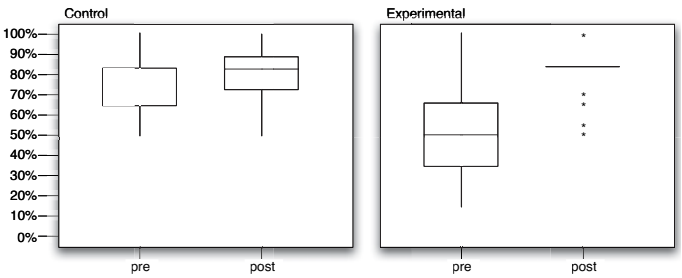

Figure 2: Threads Test Results.

This time two different instructors were teaching the classes, which might have lead to the students in the control and experimental sections having different levels of understanding. The experimental section, however, showed a 28% mean gain in one session that may have taken the other session two. A significant change occurred in the experimental section (Figure 2): the wide distribution shrunk to what appears to be a line instead of a box for the post test. Also visible is the climb in low scores. Most of the scores in the experimental section grew by more than 25%, whereas in the control section the lower scores stayed low.

Further Analysis

There are many different types of atypical techniques. Role-playing is just one example. Each technique targets different learning styles and

has a different capacity to entertain students. The role-play most likely targeted the active learners in the course who were not learning as effectively due to the high frequency of lecturing. The experimental section, although motivated and entertained by the activity, may not have been the most receptive to the style of active learning.

In both experiments, the students who first scored below their section's average (lower-half students) were able to make significant gains in scores after a role-play. The lower-half students in the experimental section showed more increase than those in the control section (Figure 3). This shows how the role-play exercises benefit the students who began with less than average mastery of the CS concepts.

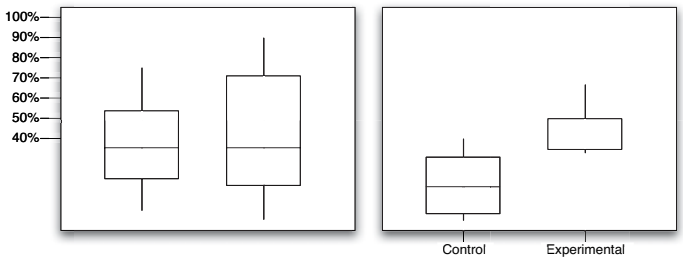

Figure 3: Increases for students who started below average.

While not all the students benefited from the use of these atypical classroom techniques, the average gain in lower-half students is much larger than the average gain for the whole class. This might imply that the students who began above the average (upper-half students) did not benefit from the role-play and perhaps actually suffered from such a concentration on active learning. This could be one reason active-learning techniques are not implemented more frequently. In their paper on active learning, Bonwell and Eison write that "perhaps the single greatest barrier of all, however, is the fact that faculty members' efforts to employ active learning involve risk: the risks that students will not participate, use higher-order thinking, or learn sufficient content, that faculty members will feel a loss of control, lack necessary skills or be criticized for teaching in unorthodox ways" [1]. The experiment shows that the role-play had a positive effect on the students who were struggling in the course and minimal negative effects on the students who were doing well with the lecture and group-work methods.

Conclusion

In summary, atypical techniques can be used to help instructors become more effective in teaching their students introductory concepts in computer science. Not only can professors make CS concepts more fun, but they can also pique students' curiosity, cater to their learning style, and relate ideas in new and interesting ways. The use of unorthodox methods in the classroom helps an instructor to interest the class, and more importantly, keep the class interested. By gaining the students' attention they can reiterate concepts without seeming repetitive, and by varying teaching methods instructors can target the various learning styles needed in the classroom. As the University of Washington witnessed with their own Steve Wolfman, these atypical techniques can be used to make computer science courses more effective and interesting for both student and teacher.

Acknowledgments

Many people were involved with the construction of this paper. The long list of people includes, but is not limited to the following:

- *Dr. Mark A. Ardis helped me construct a plan for my research and writing, and provided direction to me as a new student in the field of CSE.*
- *Dr. Andrew Kinley helped me construct and implement my experiments on the students in CS120.*
- *Dr. Cary Laxer allowed me to conduct quizzes in his classroom and provided support and feedback for my experiments.*
- *Dr. Gwen Lee-Thomas helped me to design and think through the implementation and evaluation of my experiments.*
- *Steve Wolfman helped point me to good sources of information on Computer Science Education, and provided me with a prime example of implementing atypical techniques in CS classes.*
- *Dr. Caroline Carvill, Dr. David Finn, Dr. Andy Kinley, Dr. Daniel Morris, Dr. Laurence Merkle, Dr. Mark Yoder, and Dr. Arthur Western let me interview them about their teaching techniques and general classroom practice.*
- *Dr. Gary Sherman, Dr. Carvill, Dr. Kinley, and other professors provided me with the inspiration to see how atypical techniques would affect CS education.*

References

1. Bonwell, C. C. and Eison, J. A. 1991. *Active Learning: Creating Excitement in the Classroom*. ERIC Clearinghouse on Higher Education.
2. Bradley, M. *Guess Who?* (Game).
3. Buckland, R. 1996. Can we improve teaching in computer science by looking at how English is taught? In *Proceedings of the 2nd Australasian Conference on Computer Science Education*, ACM Press. 155-162.
4. Felder, R. M. 1996. Matters of style. *ASEE Prism* 6, 4. 18-23.
5. Harrill, R. 2002. University of Washington Recognition Awards 2001-02. *UWeek Awards*.
6. Matocha, J., Camp, T., and Hooper, R. 1998. Extended analogy: an alternative lecture method. In *Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education*, ACM Press. 262-266.
7. Wolfman, S. A. 2002. Making lemonade: Exploring the bright side of large lecture classes. In ACM SIGCSE Bulletin. 257-261.

Biography

Sid Stamm (sstamm@indiana.edu) is a Ph.D. Student at Indiana University and recently graduated from the Rose-Hulman Institute of Technology with a BS in Computer Science. He is interested in functional programming languages and Computer Science Education as it specifically relates to entry-level studies for people who may not be pursuing Computer Science as a career or might not yet be interested in the field.

This article originally appeared in *Crossroads* 10.4 (Summer 2004), "Computer Science Education."