article:1145

Making Visible the Behaviors that Influence Learning Environments

Research and theory suggest that women are retained at lower rates than men in computer science (CS) because of lack of role models, a male-oriented curriculum, loss of confidence, and an educational culture oriented toward males or even “hostile” toward women. We studied undergraduate CS classroom behavior hoping to contribute to improved education for all students, not just women. We framed the study with theories of learning that take into account the social, cultural, and physical aspects of an educational setting (socio-cultural perspective, learning environment, and communication climate). In educational settings, a supportive climate is one where students can make mistakes without fear of embarrassment or loss of social status.

Using triangulated ethnographic inquiry, we observed 348 hours of mainly first- and second-year courses and other departmental functions, conducted innumerable informal interviews with students and faculty and 51 formal student interviews, analyzed student records, and reviewed documents. Appropriate techniques were used for each data type (e.g., statistical comparison of quantitative data; pattern analysis and coding for observation data).

Our findings revealed that in spite of instructors’ thorough preparation and desire to encourage student interaction, the CS classrooms observed tended toward a “defensive climate.” Classroom climates tended to be impersonal. Even in small classes, faculty tended not to use student names and students tended not to have typical student conversations. The impersonal nature of classrooms was reinforced by a fear of working with other students. To ensure that each student acquired the knowledge needed to advance to the next level and to avoid cheating, the department had a strict policy with respect to plagiarism. A computer program was used to grade students’ programs and could identify cheating. Syllabi contained warnings that unintentionally aligned the word “collaboration” with “plagiarism,” leaving many students confused about whether they were allowed to talk to other students about their homework. In interviews, students corroborated this fear

No one *has* to take CS in high school, so as a result, the range of experience in an introductory programming course is wide and extremely problematic for professors. Many times, we heard professors talk about the differences in experience where they inadvertently aligned experience with intelligence. For example, they might say that there were people who had never programmed, people who had done some programming, and “rocket scientists,” a term commonly used to refer to highly intelligent people. As a result, students learned quickly that displaying their experience could be a bid for higher status than that of their peers. In nearly every class period, certain students, usually male, would ask questions or use jargon that

seemed intended to impress, rather than seek information. Interviews revealed that as a result of this kind of behavior, inexperienced students felt that they did not know enough to do well in the course. To avoid embarrassment, many stopped asking questions in class. In addition, in an effort to encourage students, professors and teaching assistants sometimes told classes that the test or assignment would be easy or that everyone would get 100%. Unfortunately, this was rarely the case, so students who fared worse than their peers easily lost confidence.

Students who understood programming code better than their peers often sometimes made public corrections of professors' trivial mistakes. While the importance of accurate code is indisputable, professors were usually teaching higher-level concepts rather than the code itself. Instead of stemming the critiques, professors would validate students' right to correct them during class by making corrections in their slides. We did not observe any professor telling students that the focus was on theory and concepts or suggest to students that they come to office hours to "debug" their slides.

Taken together, the behaviors we observed encouraged a classroom climate characterized by defensive communication. The unchecked jockeying for superior status among students implied constant evaluation, aroused feelings of inadequacy, and made it likely that only highly achieving or experienced students felt comfortable revealing their knowledge. Announcements that a test would be easy, but wasn't, denied the legitimacy of students' fears that the material was difficult. The limited interpersonal interaction and fear of cheating made it difficult for students to develop the kind of learning communities and relationships needed in this difficult disciplines. Students had few opportunities to hear each other articulate what they were learning, were afraid to ask even simple questions in class (e.g., what was the average grade on the test?), and many came to believe, wrongly, that they didn't belong in this setting (mainly because of lack of information to the contrary).

Women are especially negatively affected by this classroom environment. Research shows that women tend to enter the major with less experience than their male peers, lose confidence more easily than men (even when their grades are the same), and are more critical of their grades than men (e.g., when a woman gets an average grade, she is more likely to see herself as having done poorly than when a man gets an average grade).

CS faculty are very concerned about the quality of student learning and the need to bring students from novices to full members of the discipline. Students bring many of these behaviors into the classroom with them, based in years of socialization and the varying experience they have with computing. When faculty understand how these student behaviors affect the confidence and ultimately, the learning and performance of other students, they can take control of these influences. For example, they can ask students to explain the comments and jargon they produce to the rest of the class, can use assignments that teach computing concepts that

either do not use actual programming code (e.g., using objects as commands and sequencing them) or use languages that students are unlikely to have experienced, and can create informal situations that generate conversation – particularly ones that permit students to hear each other articulate what they are learning, but that do not lead to bad feelings between students (e.g., ungraded classroom activities or informal explanations of programming choices in front of the class). Some of these suggestions are quite difficult and require creativity; in many cases, faculty need explicit institutional support for attention to teaching rather than just research.

This material is based upon research supported by the National Science Foundation under Grant No. 0090026. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Author 1: Lecia Barker email: Lecia.J.Barker@colorado.edu

Author 2: Kathy Garvin-Doxas email: garvindo@ucsub.colorado.edu

: Back to Fall 2005 Issue Vol. 1, No. 2

: Back to List of Issues

: Back to Table of Contents