Ubiquity, an ACM publication
October 2010

Cheating in Computer Science

by William Murray, CISSP

Editor's Introduction

Many computer science teachers are very concerned about students cheating in their courses. Surveys report that almost three-quarters of high school students admit to cheating within the past year.

John Barrie, founder of the plagiarism-detecting Web site Turnitin.com, says that about a third of the papers submitted to the site have significant levels of plagiarism. Many people say that the Internet has made cheating easier and harder to detect and they wonder if the moral fabric of our youth is fraying.

In the trenchant analysis below, Bill Murray approaches the teaching-learning system as a game in which students, teachers, and others play various roles. He wonders whether the game itself encourages cheating, and suggests that teachers could restructure the game so that cheating is less rewarding and less likely.

Peter J. Denning
Editor

Ubiquity, an ACM publication
October 2010

Cheating in Computer Science

by William Murray, CISSP

Nice people do not invite small children to lie, cheat, or steal or troops to mutiny.

A recent report[1] suggested that there are more academic honor violation accusations among computer science students than in any other single academic discipline. While one or two suggested systemic problems, the somewhat defensive response from computer science faculty seemed to suggest that most of these violations take the form of plagiarism, i.e., submitting someone else's computer program as one's original work. They contend that there are no more such violations in CS than in other disciplines, but rather that they, the faculty, are simply more clever at detecting plagiarism, mostly through "automagic" methods, than are other faculty. Said another way, CS students do not plagiarize more than other students but, to the extent that they plagiarize, they are more likely to be caught and accused.

I propose to respond to the problem suggested by the report without addressing the question of whether or not CS students plagiarize more or less than students in other disciplines, or whether or not CS faculty are more or less clever than other faculty. These questions avoid the fundamental problem. I propose to address the underlying problem of cheating.

I tend to see "cheating" as a symptom that something is wrong in the system, in pedagogy. Students tend to see it, not so much as a matter of honor but, as a game. "Clever" faculty are simply good at the game. Clever faculty tactics simply confirm the student's view.

When I was in prep school, Latin was mandatory but it was defined as cheating to use an interlinear translation. Indeed most of the tools that are routine in teaching language today were considered cheating then. Learning was not the issue, difficulty was the issue. Learning "should be difficult" and things that made it easy were "cheating."

Of course, most learning takes place outside school, is easy, and efficiency is the yardstick. Where efficiency is the measure, things like "code re-use" are rewarded, not penalized. Code re-use is a valuable skill not easily taught. Similarly, since most production coding takes place in teams, cooperation and collaboration should be rewarded. Restricting them teaches that the

[1] http://newsletters.networkworld.com/t/4627377/258483957/104243/0/

game is more important than the lesson.

I suggest that we have gotten the cart before the horse. We are less concerned with whether students learn the right thing than whether they learn in the way that we rely upon to measure how well they learn when compared to their peers. We do this without even having considered whether the measurement is even useful, much less necessary or even counter-productive. We do it without considering whether encouraging sin is a good way to teach morals. We do it for no better reason than tradition, habit, and inertia.

As a student, I often wondered why everything in school seemed so hard and everything in real life comparatively easy. I wondered why no one ever really taught me anything but expected me to learn it on my own. I wondered why no one ever taught me the skill of learning but expected me to intuit it from trial and failure. They persisted in this strategy even when failure became so routine as to suggest that intuition had failed me, even when failure led to discouragement and more failure. Discouragement was seen as a character flaw to be punished. "The beatings will continue until morale improves."

Note that no one ever really expected me to learn to read, write, or speak Latin. Fluency in Latin was not what it was about. It really was about discipline. Latin was useful for developing rigor and discipline precisely because it was abstract, difficult, and otherwise useless. We are still teaching computer science as if it were about programming and programming as if it were a language, like Latin, taught more for grading than for developing knowledge and skill.

At some level or another students suspect all of this. If school is more about grades and grading than about learning, then "code re-use" is the right strategy. If the simulation has only "no-win scenarios," then change the simulation.

I used to teach programming by teaching a language, vocabulary and grammar, and then saying to the student, write a program that does this, that, or some other thing. I expected the student to compose and required originality. I expected the student to respond to a specification so incomplete that one would never tolerate it outside academia.

I no longer teach programming by teaching the features of the language and asking the students for original compositions in the language. Instead I give them programs that work and ask them to change their behavior. I give them programs that do not work and ask them to repair them. I give them programs and ask them to decompose them. I give them executables and ask them for source, un-commented source and ask for the comments, description, or specification. I let them learn the language the same way that they learned their first language. All tools, tactics and strategies are legitimate.

As a teacher, my job is to help students learn, not create artificial barriers to learning in the name of equitable grading. Nice people do not put others in difficult ethical dilemmas. Grading should be a strategy for making learning more satisfying by demonstrating accomplishment. Many students demand grading so that they can measure themselves against their competitors. I cannot remedy the fact that society has turned grades and grading into something else, but I do not have to get so caught up in that agenda that I lose sight of my goals.

It is my job to satisfy the student, the paying customer, not the other way around. Perhaps cheating is a strategy that students bring to an artificial problem that I have created. Perhaps it is a symptom of my failure to teach. I may not be able to compensate for the culture and life experience that might predispose them to cheat but I should not make it worse. If one of my students responds to conditions that I have set by behavior that violates the code of honor, then I am guilty of a grave offense. If there is a contest between us, then we both just lost the game.

In the movie *Star Trek*, real life intervenes in the academic trial of Cadet James Tiberius Kirk for cheating. The intervention comes just in time to prevent an "academic" offense from destroying a promising career. Real life made moot the academic trial just as young Kirk had suggested his defense and the trial ended as the young warriors rushed off to battle. I was disappointed because I was hoping that his argument, that the "no-win scenario" was itself (faculty) cheating, might have vindicated him.

Not only do I think that we should adapt our pedagogy to reduce the motive for, value of, and opportunity for cheating but we should rethink our use of academic honor codes. Honor codes were initiated in elite schools that were engaged in character formation as much as education. Few of today's schools make any attempt at or pretense of character formation; most of the few that do are military or religious institutions. In any case, one teaches character by modeling it, not by forcing the unformed to struggle through artificial trials.

Use of honor codes to resist plagiarism in CS is, at best, arbitrary and capricious. Students, raised on games, "game" the system, sometimes with tragic results. Often the code, for example those with a non-toleration clause, is at odds with a value, for example, peer loyalty, that the student has been taught from sand-pile.

At a cocktail party in Boston, I was asked by a Harvard dean what one change I would make to improve the American school system. I suggested that I would get teachers out of the credential granting business, break down the adversarial relationship between teachers and students, and get them on the same team. He was horrified. "Without the authority of the teacher to grant credentials, the classroom would degenerate into chaos," he said. I told him that as an industrial management trainer, my students graded me, rather than the other way around. Perhaps I did not have a discipline problem because I was interesting, perhaps because my students and I respected one another. In any case, we can divide pedagogy from credential granting. The British have done it with positive results.

The issue should not be whether CS students cheat more or simply are caught more. Rather, it should be whether faculty convenience justifies the ruined lives.

About the Author

William Hugh Murray, CISSP is a “Pioneer” in computer security, a faculty member at the Naval Postgraduate School, and a founder of the Colloquium on Information System Security Education (http://CISSE.org).

DOI: 10.1145/1865907.1865908