



Jim Taylor
MTC

10/26/07

David Hunter <dhunter.westmont@gmail.com>

Our "Reasoning Abstractly" Meeting this Friday

Jim Taylor <taylor@westmont.edu>

Wed, Oct 24, 2007 at 12:56 PM

To: Ray Rosentrater <rosentr@westmont.edu>, "Russell W. Howell" <howell@westmont.edu>, Patti Hunter <phunter@westmont.edu>, David Hunter <dhunter@westmont.edu>, Wayne Iba <iba@westmont.edu>, Kim Kihlstrom <kimkihls@gmail.com>, Jonathan Leech <leech@westmont.edu>
Cc: Marianne Robins <robins@westmont.edu>

Dear Fellow Teachers of GE "Reasoning Abstractly" Courses:

Thanks for agreeing to meet with me this Friday during chapel at your office building to discuss criteria for courses to qualify for the GE Common Inquiries Reasoning Abstractly area. We will also discuss potential specific student learning outcomes for these courses.

Here's what the GE Document says currently about Reasoning Abstractly courses:

Reasoning Abstractly (i.e., Philosophy, Mathematics and Computer Science) Courses satisfying this requirement focus on critical and analytical reasoning about non-empirical, abstract concepts, issues, theories, objects and structures. Students in these course should learn to understand and evaluate abstract arguments and explanations, analyze abstract concepts and solve abstract problems.

The GE "Supplemental Document" adds the following "Interpretive Statement":

Though all theoretical disciplines other than the three to which this common inquiry area is restricted involve abstract reasoning, only these latter three disciplines (a) involve a relatively high degree of abstraction and (b) employ primarily highly abstract methods and study primarily highly abstract objects. Moreover, though courses in other disciplines have philosophical, mathematical, computational and logical elements, only courses in these three disciplines make such elements their primary focus. Finally, the GE committee should not assume that every course in these three disciplines would qualify as abstract reasoning courses. Only courses in these disciplines involving attention to formal methodology (argument, analysis, evaluation, problem-solving) would be adequate. So, for instance, courses that merely summarize philosophical views would not qualify.

In another document entitled "Student Learning Outcomes for the General Education Program" the following is included in the Reasoning Abstractly category:

Students will demonstrate skills in critical and analytical reasoning. They will be able to understand and evaluate abstract arguments and explanations, analyze abstract concepts and solve abstract problems. This will include:

- distinguishing arguments from non-arguments
- recognizing premises, conclusions, and the relationship between them
- identifying the forms of arguments
- identifying and understanding fallacies
- constructing strong and valid arguments

Please reflect on these statements in preparation for our conversation.

Sincerely,

Jim

Department of Philosophy
Westmont College
Santa Barbara, CA 93108
Voice: (805) 565-6157

Fax: (805) 565-7101

 **winmail.dat**
OK

Do software engineers need mathematics?

Software engineers often proclaim that they never use any of the mathematics they learned in college. Come to that, they say they don't use much of the computer science they learned either. As a mathematician, I'll leave it to my CS colleagues to respond to the latter allegation. (At least, I would if all my CS colleagues hadn't left academia to work for an Internet startup!) As far as the use of mathematics is concerned, however, let me respond. In my view, those software engineers are dead right: They don't use their college mathematics.

Having got that off my chest, let me go on to say that they are also dead wrong. They make use of their college mathematics education every day.

There's no paradox here. It comes down -- in Clintonesque fashion -- to what you mean by that word "use." One meaning is the one those software engineers will have encountered in their math classes. For example, having learned the rule for integration by parts in their calculus class, they were then given exercises and exam questions that required them to use that rule. This is the most familiar meaning attached to the word "use", and it's the one the engineers implicitly assume when they say they never use their college math. But it's a meaning that is built on what I call the "filling a vessel" view of the way humans learn.

According to the "filling a vessel" view, education consists largely of pouring facts into our brains, and using what we have learned consists of pouring it back out. That is, dare I say it, a highly simplistic -- and erroneous -- view of education. But it's one that the education establishment (which I'm in) fosters every time it offers a course and then measures the results by setting a largely regurgitative, three-hour, written exam.

In contrast, all the evidence from several decades of research both into the way the brain works and into the learning process -- and there is masses of such evidence -- says that the acquisition of facts and algorithmic procedures are merely surface manifestations of what goes on when people learn. (We know they are surface phenomena since we generally forget them soon after the last exam is over.) The real value of education is something else. Our brains are perhaps the world's best examples of an adaptive system. When we subject the human brain to an extended educational experience, it undergoes permanent changes. In physical terms, those changes are the growth and strengthening of certain neural pathways. In functional and experiential terms, we acquire new knowledge and skills. The more repetitive the learning process, the stronger and longer lasting are those changes.

The effect of repetitive learning is nowhere more dramatic than in mathematics. Formal mathematics is at most 5,000 years or so old. That's a mere blink of an eye in evolutionary time, and certainly not long enough for our brains to undergo any but the most minor changes. Thus, the mental processes we use to do mathematics must have been acquired and in use long before the Sumerians introduced abstract numbers some time between 8,000 and 5,000 years ago. The new twist required in order to do mathematics was to bring those capacities together and use them to reason not about the physical and social world for which they initially developed through natural selection, but rather a purely abstract world of the mind's own creation.

The human brain finds it extremely hard to cope with a new level of abstraction. This is why it was well into the eighteenth century before mathematicians felt comfortable dealing with zero and with negative numbers, and why even today many people cannot accept the square root of minus-one as a genuine number.

But software engineering is all about abstraction. Every single concept, construct, and method is entirely abstract. Of course, it doesn't feel that way to most software engineers. But that's my point. The main benefit they got from the mathematics they learned in school and at university was the experience of rigorous reasoning with purely abstract objects and structures.

Moreover, mathematics was the only subject that gave them that experience. It's not what was taught in the mathematics class that was important; it's the fact that it was mathematical. In everyday life, familiarity breeds contempt. But when it comes to learning how to work in a highly abstract realm, familiarity breeds a sense of, well, familiarity -- meaning that what once seemed abstract starts to feel concrete, and thus more manageable.

Though the payoff from learning (any) mathematics is greater for the computer professional than most other people, in today's society the benefits affect everyone. For instance, a study carried out by the US Department of Education in 1997 ("the Reilly Report") showed that students who complete a rigorous high school course in algebra or geometry do much better in terms of gaining entry to college or university, and perform much better once they are there, whatever they choose to study. In other words, completion of a rigorous course in mathematics -- it is not even necessary that the student does well in such a course -- appears to be an excellent means of sharpening the mind and developing mental skills that are of general benefit.

In my recent book [The Math Gene: How Mathematical Thinking Evolved and Why Numbers Are](#)

Like Gossip I examine these ideas more fully, fleshing out just which mental capacities are required to do mathematics, and identifying possible survival advantages that led to those capacities being selected for in our ancestors. I also show how acquisition of the ability to cope with abstraction -- to marshal mental capacities developed to handle concrete objects and real-world circumstances, and apply them to abstract entities -- confers some benefits on everyone. As I have tried to indicate in this short essay, the benefits for the software engineer are far greater. Indeed, they are an essential prerequisite. That's not usually given as the "official reason" for the obligatory "math requirements" for engineering students. But it is, I suggest, the main reason why they are beneficial.
