Should Mathematics Be A Mandatory Fundamental Component Of Any IT Discipline?

Chaker Eid, University of Bahamas, Bahamas
Richard Millham, University of Bahamas, Bahamas

ABSTRACT

In this paper, we investigate whether and how mathematics factors into students' performance in IT learning. The involved cognitive levels of students learning mathematics and hence problem solving, are correlated to how well they are able to transpose their knowledge and apply it to problem solving in the IT field(s). Our hypothesis is that if students perform better in mathematics, in terms of level of mathematics course and grade earned in that course prior to engaging in IT studies, their performance in IT will also be higher. The performance of several groups of students, over a period of five semesters are collected, analyzed and the correlation between mathematics learning ability and that of IT is measured, analyzed, and the results are reported and therefore the hypothesis is tested.

Keywords: IT Learning; Mathematics Education; Cognitive Development

INTRODUCTION

The correlation of mathematical concepts with those of computer science has long been established. (Nguyen, 1997; Martin-Lof, 1982) Much research work has focused on establishing a link between mathematical achievements and programming proficiency. (Ignatuk, 1986; Cafolla, 1987) Some of the work has focused on establishing a relationship between mathematical ability, programming achievement, and Piaget's theory of cognitive development. (Cafolla, 1987; Lott, 1984) However, most of the work has focused on the relationship between mathematical and programming proficiency. Feurzig argues that rigorous thinking, the ability to express oneself clearly, and the need for precise statements is needed for algorithms and programming. (Feurzig, 1969) In this paper, we argue, as our hypothesis, that the preciseness, abstract thinking, and rigorous reasoning needed by mathematics are also needed in related information technology courses, not only programming. To validate our hypothesis, we compare the achievements of students in a diverse group of information technology (IT) courses (programming, databases, and management information systems) with their achievements in the highest mathematics course taken prior to the IT courses. Our argument is that the type of thinking (preciseness, abstract thinking, and rigorous reasoning) developed and required by mathematics can be transferred to information technology courses with a resulting increase in performance.

BODY OF PAPER

The foundations of computer science have been based on mathematics. Several researchers have established the relationship between mathematics and computer science in terms of their equivalencies in type of thinking and operations. Martin-Lof has established the equivalencies of computer programs with their mathematical equivalents. Programming functions with parameters and return values can be viewed as equivalent to mathematic functions with arguments and values. Computer data types and structures have their basis in mathematic set theory. Sequence of instructions in a program may be seen as a composition of functions where each function represents a computer instruction. Computer basic instruction structures such as if-then and loop statements correspond to cases of functions and recursive functions in mathematics respectively. (Martin-Lof, 1982) Because of the discrete nature

of objects being manipulated by computers within their programs and the discrete basis of many computer algorithms, the use of discrete mathematics is highly useful. Common abstract computer concepts such as trees, graphs, and tables are based on discrete mathematics. Discrete mathematics forms a basis for many computer science concepts and algorithms. An example, commonly computers manipulate abstract structures, such as trees, graphs, and tables, which have their basis in discrete mathematics. Many computer algorithms, such as finding the shortest path between two points, utilize discrete mathematics. However, continuous mathematics, such as calculus, is often used, not only in scientific computing, but also in determining which algorithm in a group of algorithms will run the fastest. (Nguyen, 1997)

Other researchers have found that programming requires precision and both analytical and rigorous thinking which is developed and required in mathematics. Rigorous thinking and expression are required by programmers. In order for their algorithms to be executed, programmers need the ability to state themselves precisely. (Feurzeig, 1969) "The automatic computer really forces that precision of thinking which is alleged to be a product of any study of mathematics". (Forsythe, 1959) By trying to formalize things in algorithms, we are a led to a much deeper understanding of the subject material than if we thought of the same things in the traditional way. Giving examples to demonstrate this statement, Knuth states that linguists thought that they understood languages until they tried to formalize it in computer algorithms; they then learned that they knew far less than they originally thought. Similarly, many modelers learn more about their model when setting it up for computation than they did in examining the output. By developing algorithms for high-speed numerical calculations, every theorem in elementary number theory will arise in a natural, motivated way. Using the algorithmic approach, understanding is improved in many ways. (Knuth, 1974) Phongutta , in his work, demonstrated the correlation between mathematical achievement and analytical thinking ability, regardless of a student's particular attitude toward math. (Phonguttha, 2009)

This type of thinking is linked with the highest cognitive level of thinking, as defined by Piaget. Handling of variables, relationships between variables and formal methods were found to be descriptive of the highest cognitive level of thinking, formalization. (Adley and Shayer, 1994) Cafolla has shown the relationship between mathematical ability, programming proficiency, and Piaget's cognitive development. (Cafolla, 1987) All of these areas are influenced by the left hemisphere of the brain (Lott, 1984). Using Lawson's Test of Scientific Reasoning, which includes questions on conservation, proportional thinking, identification of variables, probabilistic thinking, and hypothetico-deductive reasoning, in order to determine student's reasoning level, Maloney demonstrated that the highest test score averages reasoning level (2/3 of students) were found among students taking algebraic and calculus-based physics courses while barely 1/3 of students in health and education achieved this highest reasoning level. (Maloney, 1981)

Piaget developed a cognitive development theory composed of three states of development: pre-operational, concrete, and formal operations. (Piaget 1972 & Epstein, 1990). Pre-operation is considered to be the mental age of people during the 2 to 7 years old stage. The concrete conceptualization state includes people who are able to understand the conservation of matter and classification/generalization of things, such as all cars are vehicles but not all vehicles are cars. However, people at the concrete conceptualization state, according to Barker, are unable to understand concepts such as mathematical ratios. (Barker, 1983). The highest cognitive development level in Piaget's theory is formal operations. Formal operations are the ability of people to deal with abstractions, form hypotheses, solve problems systematically, and engage in mental manipulations. (Biehler and Snowman 1986) In order to develop the ability for formal operational thinking, the ability for biconditional reasoning ("if and only if" logic) is required first. If the person cannot utilize biconditional reasoning, they would be unable to make the move from concrete to abstract/logical thinking. (Chiapetta 1976) Without the ability for abstract/logical thinking, learning procedural programming is difficult (Becker, 1982). However, research has indicated that formal operations, such as abstract and logical thinking, may occur much later in some people or never occur (Griffiths 1973, Schwebel 1975, and Pallrand, 1979).

Table 1. Programming Languages and Cognitive Development/Style Programming

	Piaget's Cognitive Development Levels Cognitive Style				
Paradigm	Pre-Oper	Concrete	Pre-Formal	Formal	Formal (Hemisphericity)
Procedural (COBOL, logic sequence)	Burnout	Burnout	Burnout	P & M	Left Brain
Object Oriented (C++, Java, concepts)	Burnout	Burnout	Burnout	P & M	Either hemisphere
Visual (Visual Basic)	Burnout	Burnout	P & M	Bored	Either hemisphere
Script (HTML Web pages)	Burnout	P & M	Bored	Bored	Right Brain

(P & M; productive and motivated) (White, 2002)

Consequently, many researchers have that students require a strong mathematical background for programming. Ignatuk has shown that a strong mathematic background is a predictor of success in procedural programming. (Ignatuk, 1986) Saiedian uses the necessity of a strong mathematical background for success in programming as his argument for mathematical courses being prerequisites for programming. (Saiedian, 1992) Eid found that having students learn the basics of programming, procedural programming which in itself heavily based on mathematics, produces better performance in advanced visual programming classes than if students first took an introductory visual programming class. (Eid and Millham, 2011)

Research by Bennedsen could not find a correlation between math scores and programming scores. The lack of correlation was attributed to the correlation being based on the use of a single binary (pass/fail) practical test at the end, which may not have measured the student's success well (Bennedsen and Caspersen, 2006) in our study, our final scores are a compilation of student scores on assignments, projects, and several exams throughout the course and, thus, these scores are more reflective of the student's entire work during the course than a single pass/fail exam at the end.

Our research utlised a correlation of a student's highest prior math course performance with their performance in an IT class. Their performance in the highest math course prior to taking selected IT courses divided students into two groups, for comparison purposes, for each IT course. Students achieving a grade 65% or better in their highest math class, calculus, would be placed into one group while students achieving less than a grade of 65% were placed in another group. The cutoff grade of 65% was set because performance at this grade level or above is required of students majoring in computer science. Our findings indicated that if a student has taken calculus prior to taking an introductory programming course, their performance in this calculus class is correlated with their performance in the introductory programming class. If the student had achieved a grade of 65% or better in the calculus class, their grade in the introductory programming class averaged 67%. Their standard deviation was 4.0. If the student had achieved a grade of 64% or below in the calculus class, their grade in the introductory programming class averaged 53.5%. Their standard deviation was 7.1, a significantly higher deviation.

Similarly, if a student had taken calculus before taking an advanced programming class and achieved a grade of 65% or better, their average grade in the advanced programming class was 72.5%. Their standard deviation was 2.5. If their calculus grade was below 64%, their average grade in the advanced programming class was 60%. Their standard deviation was 3.1. The greater difference in average grades could be explained by the fact that the advanced programming class involved a higher level of abstract thinking involving classes, inheritance, polymorphism, and databases.

Although there is an established correlation between prior mathematical background and performance in regards to programming, whether procedural or visual, what is the relationship between mathematics and non-programming courses? Other IT courses, such as databases, have a mathematical basis. Relational databases, the most common currently used database model, is based on Codd's mathematical work that uses relational algebra and calculus to describe database structures and their manipulation. (Codd, 1970) Databases rely on data modeling for their structure. Data modeling, which includes Entity-Relationship Diagrams, can be expressed using the mathematically-based Descriptive Logics to represent the essential features of the data model and to provide a series of powerful reasoning techniques to derive information from this model. (Chomicki and Saake, 1998)

In our study, we also measured the performance of students in databases in regards to their grade in a prior calculus course. If a student achieved a grade of 65% or better in calculus prior to taking the advanced database

class, their average grade in the advanced database class was 76.5. If the student's calculus grade was less than 65%, their average grade in the advanced database class was 52.3%.

But what about performance in IT classes with no direct basis in mathematics, such as a general information management systems class? Performance in a prior calculus class also applies to non-programming and non-database classes, as well. If a student's grade in a prior calculus class was 65% or better, the average grade in an information systems management class was 72.1%; otherwise, if they had a calculus grade below 65% prior to taking this information systems class, their average grade in the information systems class was 59.8%. Based on these findings, it appears that prior mathematics learning improves a student's performance in a general IT course. Management Information Systems is a non-technical and non-mathematically based course but it involves extensive problem solving tasks and it often entails making a series of decisions. The problem-solving ability and rigorous systematic thinking, which is developed in a prior mathematics course, may be transposed to solve problems in management information systems.

CONCLUSION

The correlation of the performance of two groups of students (those that achieved 65% or above in their highest mathematics class of calculus and those that achieved less than 65%) with their performance in different IT courses have been confirmed by our findings. Our hypothesis was that the preciseness, abstract thinking, and rigorous reasoning required by mathematics courses is also required in different types of IT courses (not just programming courses). As part of our approach, we divided up students into two groups based on their performance in the highest level mathematics course, calculus, which they took prior to taking selected IT courses, and compared their performance in these IT courses with their prior performance in mathematics. Regardless of the type of IT course (programming, databases, and management information systems), students who achieved a grade of 65% or better in calculus performed significantly better in the IT course than those students who achieved a lesser calculus grade. These findings validate our hypothesis. By being required to think abstractly, precisely, and rigorously in mathematics, students are able to achieve a higher cognitive level, as defined in Piaget's cognitive development theory, which enables them to transpose their mathematical concepts to help them to grasp the abstract concepts of IT, whether mathematically based or not, and apply them to more concrete concepts, if necessary. If students do not achieve this higher level of cognitive development, through more than minimal performance in prior calculus courses, they tend to remain at the lower concrete level. Consequently, they are unable to think in an abstract, precise, and rigorous way with a resulting loss of grade performance in subsequent IT courses. We believe that mathematics indeed needs to be a mandatory component of any IT discipline, not just computer science.

AUTHOR INFORMATION

Richard Millham is an assistant professor of computer science at the University of Bahamas and a visiting research professor at Durban University of Technology. After working in industry for 15 years, he joined academe and has taught for many years. He has worked revising curriculum for the University of Bahamas, Catholic University of Ghana, and the Catholic University of the Sudan. His teaching interests are in the area of software engineering and evolution, service oriented computing, creative computing, and programming. He holds a BA(Hons) in Computer Science from the University of Saskatchewan, MSc from the University of Abertay in Dundee, Scotland, and a PhD from De Montfort University in Leicester, England. Contact information: Professor Richard Millham, University of Bahamas, School of Business, COB, Box N4912, Nassau, Bahamas. E-mail: richardmillham@hotmail.com (Corresponding author)

Chaker Eid is an assistant professor of computer science at the University of Bahamas. He has over ten years of IT industry experience supplemented with eleven years of IT/mathematics teaching experience. He holds a BSc in Mathematics from the University of Windsor in Canada and two MScs, one in mathematics and the other in computer science, from the University of Detroit-Mercy in the US. Contact information: Professor Chaker Eid, University of Bahamas, School of Business, COB, Box N4912, Nassau, Bahamas. E-mail: chakereid@yahoo.com

BIBLIOGRAPHY

1. Adey, P and Shayer, M. [1994], *Really raising standards: cognitive intervention and academic achievement*, Routledge, London, England.
2. Barker, R. J. and E. A. Unger [1983],"A Predictor for Success in an Introductory Programming Class based upon Abstract Reasoning Development." Proceedings of the 14th SIGCSE Technical Symposium on Computer Science Education of the ACM, Orlando, Florida.
3. Becker, H. J. [1982], "Microcomputers in the Classroom -- Dreams or realities?" ERIC(ED217872).
4. Bennedsen, J.B. and Caspersen, M.E.[2006]. "Abstraction Ability as an Indicator of Success for Learning Object-Oriented Programming?", *SIGCSE Bulletin inroads*, Volume 38, Number 2, June 2006, pp. 39-43.
5. Biehler, R. F. and J. Snowman [1986], *Psychology Applied to Teaching*. Houghton Miffin Company, Boston.
6. Cafolla, R. [1987], "The Relationship of Piagetian formal Operations and other cognitive factors to computer programming ability (Development)." Dissertations Abstracts, A47(7), 2506.
7. Chiapetta, E. [1976], "A review of Piagetian studies relevant to science instruction at the secondary and college level." *Science Education*, 60, 253-261.
8. Chomicki, J., G. Saake [1998], Logic for Databases and Information Systems, Kluwer, New York.
9. Codd, E.F. [1970], "A Relational Model of Data for Large Shared Data Banks". *Communications of the ACM 13* (6): 377–387.
10. Eid, C, R. Millham [2011],. "Which Introductory Programming Approach Is Most Suitable For Students: Procedural Or Visual Programming?", *American Journal of Business Education*, Littleton, Col., USA.
11. Forsythe, G. E. [1959], "The role of numerical analysis in an undergraduate program" *The American Mathematical Monthly*, 66, pp. 651-662.
12. Griffiths, D. H. [1973], "The Study of the Cognitive Development of Science Students in Introductory Level Courses." ERIC(ED096108].
13. Ignatuk, N. [1986], "An Analysis of the Effects of Computer Programming on Analytical and Mathematical skills of high school students." Dissertation Abstracts, A47(3), 854.
14. Feurzeig, W., et al [1969], "Programming-Languages as a Conceptual Framework for Understanding Mathematics", Final Report of the Logo Project.
15. Knuth, D.E.[1974], "Computer Science and Its Relation to Mathematics", *The American Mathematical Monthly*, vol. 81, no. 4, pp 323-343.
16. Losh, C. L. [1984], "The relationship of student hemisphericity to performance in computer programming courses." Dissertation Abstracts A44(7), 2127.
17. Maloney, D.P. [1981], "Comparative reasoning abilities of college students," *Am. J. Phys*. 49(8), pp. 784-786.
18. Martin-Löf, P. [1982]. "Constructive mathematics and computer programming." In *Logic, Methodology and Philosophy of Science* VI. Eds. Cohen, et al. North-Holland, Amsterdam. pp. 153–175.
19. Nguyen, H.T., Krienowich, V. [1997], *Applications of Continuous Mathematics to Computer Science*, Springer, New York, USA.
20. Pallrand, G.J. [1979], "The transition to formal thought." *Journal of Research in Science Teaching*, 16, 445-451.
21. Phonguttha, R., S. Tayraukham, P. Nuangchalerm [2009]. "Comparisons of Mathematics Achievement, Attitude towards Mathematics and Analytical Thinking between Using the Geometer's Sketchpad Program as Media and Conventional Learning Activities", *Australian Journal of Basic and Applied Sciences*, 3(3), pp. 3036-3039.
22. Piaget, J. [1972], Intellectual evolution from adolescence to adult. *Human Development*, 15, 1-12.
23. Saiedian [1992]. "Math of Computing." *Computer Science Education*, 3(3), 203-221.
24. Schwebel, M. [1975], "Formal operations in first year college students." *Journal of Psychology*, 91, 133-141.
25. White, G. L. [2002], "A Theory of the Relationships between Cognitive Requirements of Computer Programming Languages and Programmers' Cognitive Characteristics" *Journal of Information Systems Education*, Vol 13(1), pp 60-66.
26. White, G., M. Sivitanides [2003], "An Empirical Investigation of the Relationship Between Success in Mathematics and Visual Programming Courses", *Journal of Information Systems Education*, Vol. 14(4).

NOTES