Turning math attractive to computer science students: An application-to-model approach

Aleardo Manacero Jr. and Norian Marranghello
Dept. of Computer Science and Statistics, Unesp
São José do Rio Preto, Brazil - 15054-000
aleardo,norian@dcce.ibilce.unesp.br

Abstract

This paper describes an innovative approach to develop the understanding about the relevance of mathematics to computer science. The mathematical subjects are introduced through an application-to-model scheme that lead computer science students to a better understanding of why they have to learn math and learn it effectively. Our approach consists of a single one semester course, taught at the first semester of the program, where the students are initially exposed to some typical computer applications. When they recognize the applications' complexity, the instructor gives the mathematical models supporting such applications, even before a formal introduction to the model in a math course. We applied this approach at Unesp (Brazil) and the results include a large reduction in the rate of students that abandon the college and better students in the final years of our program.

1 Introduction

Every year many young students come to undergraduate programs on computer science or computer engineering with very different expectatives. One of them is that they will spend their time almost exclusively in computer labs and that no extra knowledge on mathematics is necessary. This expectation is promptly frustrated by several math courses in the freshman year, which are followed by some other math afterwards, all of them mostly recommended by serious professionals associations like ACM and IEEE's Computer Society [?] or the Brazilian Computer Society [?].

From this frustration, some students usually do not apply the needed effort on the math subjects. That leads to two different situations: the student abandons college or has some trouble on understanding computer topics where that math is demanded. Neither of these situations should be neglected by the faculty body, and actually they are not, since they mean a high cost for the society, either through an unqualified professional or through an unsatisfied former student.

Although most colleges have some technique to overcome this problem, they usually restrict their intervention on asking math lecturers to give more examples oriented to computer science applications. Other typical solution is the introduction of a design course in the freshman year, as the coalition colleges do in the USA (two reports on some of these experiences are found in [?] and [?]). Unfortunately neither technique has proven successful as a whole, partially achieving their goals.

This paper describes a different approach to deal with freshman students of the Computer Science program from Unesp at Rio Preto. It is part of our curriculum since 1996 (as a pilot experience in 1996 and as a regular course since then). It is based on an application-to-model strategy, i.e., the learning of math is stimulated first by the learning of a computer application that demands that math. This is performed inside an introductory course given by lecturers from the computer science department, who have the needed knowledge on both the application and the math.

The following section reviews some of the problems related with math courses for Computer Science students, with a special emphasis on abandoning rates and repeated failures in the freshman and sophomore courses. After that we have a section describing the conception of the application-to-model approach adopted here. This approach is better understood through some examples contained in our course program, which is found in a separate section. Finally, we include a section detailing the results achieved and drawing some conclusions about this experience.

2 Teaching math: reasons and problems

Among all courses taken by undergraduate students in the technical fields the math ones are unanimously considered as the hardest. This common conclusion comes from several aspects, mostly due to the basic nature of math. In the computer science field the situation is exactly the same, if not worsened from the popularization of computers as appliances. Unfortunately, all the math taught at college is demanded in order to produce a good professional after undergraduate school.

Before listing the problems that appear while teaching math, it is reasonable to show why it is part of computer science curriculum. Primarily, one can focus this discussion on the definition of computer science, i.e., distinguish between a computer user (in its various levels) and a computer scientist. While the former is concerned with computers as a tool for other users, the latter is concerned with computers by themselves. Therefore a computer scientist aims at the development of the science, including techniques and tools that will be applied by users in their problems.

To become such professional one has to master systems modeling, demanding a strong basis on mathematics, specially on statistical models, linear algebra, propositional calculus and discrete mathematics. The knowledge's breadth and depth in any of these areas depend on the particular subject of computer science under consideration. For example, computer graphics needs more linear algebra than propositional calculus, while compiler design needs the opposite. Under any circumstance, it is impossible to know in advance which kind of mathematics a future professional will need, turning mandatory a broad range of topics inside the math courses.

Since a good curriculum has to keep a considerable amount of math courses to provide a strong basis to its students, all students have to go through several of such courses, some mandatory, some elective. The elective courses do not mean trouble for the students because only those really interested in their subjects will apply for them. On the other side the mandatory courses are not an option and causes most of the trouble to freshman students.

The causes for this troubles are:

- Computers are much more attractive than math books;
- Students cannot devise applications for theoretical subjects in math;
- Math lecturers usually do not have time and knowledge to find computer applications for every subject;
- When the lecturers know such applications, their students do not have confidence about the applications' descriptions since they were made by a non-specialist.

These difficulties lead to a rise in the number of students that leave the course during the first year. It is usual to verify abandoning rates going over 30%. Although this is not a phenomenon restricted to computer science courses it does cause a large impact in the college administration since the laboratories will have to account for an irregular demand and the number of accepted students must be higher than the social and economical best value.

Another problem is a process of repeated failures to get an approval in the basic courses, mainly because the students do not feel comfortable with math and do not apply the needed effort. The consequence of these failures are a delay in the course completion, rising substantially the educational costs and the frustration among students.

Unfortunately these are not the major problems introduced by a bad basis on mathematics. The biggest problem is the lack of knowledge on such subjects when students (or even professionals) are challenged by applications that use math in a broad sense. At that moment one student has two different paths to follow: go back to math books and learn the subject autonomously or try to learn the application only enough to surpass the current problem. While the former is the most indicated it is the latter that is usually adopted, leading to superficial analysis and specifications, or in other words to poor software.

3 The application-to-model approach

As stated before, almost every college does have some mechanism to reduce the problems described in the previous section. These mechanisms are focused on two different approaches, one that leaves to the math lecturer the responsibility of introducing computer applications to justify the mathematical topics, and other that brings small design courses to the freshman year (as the coalition colleges do).

Although the former strikes the problem directly, it does not have much success since either the professor is not used with the application or the students do not trust in the professor's knowledge about computing. The latter strikes mainly the abandoning rate, trying to stimulate students with an application but it does not apply much effort on the need of math to the correct development of such application.

Here we propose a different approach, which is partially based on the coalition approach, that is, the idea behind it is to bring to freshmen, through a one semester course, some computer applications as design examples, centering the understanding of each application into the mathematics needed to develop it. Since the instructor(s) will be computer scientist(s) the problems with confidence on the application's knowledge are reduced or

Table 1: Course's main structure.

Section	Topic	Application Introduced
1	Computer Science	——
2	Analytical Geometry	CAD, computer vision
3	Calculus	Image processing, performance analysis
4	Linear Algebra	Modeling, data packets' flow optimization
5	Numerical Analysis	Computer graphics

eliminated. In the same way, each student should be able to identify a certain area of computing that would be of interest, verifying that the work on that area can be more efficient if one applies a strong knowledge of mathematics, what improves the stimulus to learn math.

As indicated at the beginning of the previous paragraph, the technique adopted is to provide some piece of software to the students, who will play with it for a short while. After that, the instructor explains the technology used to develop such application, pointing out aspects of it that could not be implemented without a certain topic of mathematics. Through this strategy, the students are leaded first to find that the application is challenging and, later, to discover a method (the mathematical model) that solves efficiently the design issues involved with the application.

Finding math an useful tool for computers results in a greater interest on some subjects, that could be better studied at the right time. As an example, one would pay more attention to set theory if he/she knows that computer languages are mostly structured around first order logic, which is a direct subproduct of set theory. This observation leads to the fact that the applications (and this course, by consequence) should be introduced before, or at least at same time as, the math topics.

When to insert this course is the most severe restriction since it is useful only if scheduled very early in the curriculum, that is, in the first semester of the first year. At that moment we may direct the students attention to the math courses given in the same semester, and those thereafter too. If this course is left for a posterior moment we may take advantage of a higher understanding of math subjects but some of these subjects would be already left behind during the previous courses, voiding the stimulus effort.

From this earliness, one may argue that this approach cannot be developed in its totality because a freshman does not know all the mathematical subjects needed to fully understand the application. However, this is true only if the lecturer demands deep details of the application or the math involved with it, what is not actually necessary at that moment, or in other words, the math should be left to the math instructor and the application details should be left to a later course inside the curriculum. Therefore, the instructor has freedom to choose what to do in each lecture.

The choice of the set of applications used as examples and the depth of the subjects introduced with them are left to the instructor. The applications' choice are not a real trouble during the course since there are many options and we can adopt those that we are more familiar with. From the students perspective there are no restrictions on learning one application or another, except, maybe, by a higher interest in one of them. The depth's choice, on the other hand, is really critical.

Choosing the right depth of the mathematical subjects introduced with every application is critical since the success of this approach depends of the student's interest on each topic, what may be drastically reduced if the application would not be minimally understood due to the mathematics underneath it. Next section includes two examples that have been very successful in our experiments but, just as a guideline to what may be introduced in this course, no dense theory must be explained in its full formalism. The instructor must remember that he/she will be explaining theory of probabilities to students that do not know how to integrate a simple polynomial function for example. The topics should be learned mostly in an informal form, leaving the formalism to the math professors, who are responsible, and better prepared, for this task anyway. Besides that, excessive informalism must be avoided since it could mislead students to believe that that math is easier than it really is, what would result in the same damages described in the previous section.

4 Course description

The application-to-model approach described here was officially implemented in the Unesp at Rio Preto's Computer Science course in 1997, although a pilot experiment was conducted in 1996. Historically, it was intended as an introductory course that would provide an overall view of computer science, giving to our students conditions to understand what computer science really is and what can be done by a computer scientist. During the conception of this course we noticed that it could also be used as a tool to stimulate the learning of math, what leaded to this mix of goals since its trial version in 1996.

During these years we could develop a better understanding on what topics are more comfortable to the students and what applications they like to play with. From these experiments we divided our course into five sections, as shown at Table ??. Four sections deal explicitly with some application and the mathematics applied to enable each one. The initial section does not contain any mathematics or applications and the reasons behind this are twofold: first the students may have some time to get acquainted with two math courses (vectors and analytic geometry and a first course on calculus) and second we have some time to show them the differences between the various flavors of programs related with computers (information systems, computational science, computer science, etc), how our program fits into this and how are the career perspectives.

During the other four sections we usually introduce more than one application in order to give the students the notion that math is embedded everywhere. Since the time is reduced (only two hours/week), not all of these applications are carefully examined, some of them appear in the section only after the math could be explained through some other application, just as an alternative case. As an example, in the section that deals with linear algebra we start showing the problem of transmitting data packets over a complex WAN (Wide Area Network) in an efficient manner. After that we show that a technique provided by optimization theory (network flows) can solve this problem with a high precision. Then the students are ready to learn how this technique works, leading to the need of linear algebra, which is introduced by then. When the students understand the relevance of linear algebra for computer science we give a brief introduction to systems modeling with Petri nets and how a Petri net could be represented and analyzed by matrices.

In the following lines we provide some details about how sections two and three are developed during the semester. These descriptions are not complete since the applications choice and details are not a restriction of our approach, what turns the descriptions made here simple indicatives on what can be done in this course.

4.1 Introducing analytical geometry

After the introductory section the students are faced with an attractive application, a CAD system. The procedure followed by the instructor at this point is summarized below:

1. Demonstrate a CAD session, making the students draft a simple part;
2. Describe its functional components, emphasizing the treatment given to draft design. Explain also functions like movement simulation and mechanical analysis;
3. Develop the feeling that geometrical representation (points, lines, curves and surfaces) is better than pixel representation when both are applied to CAD, what is performed from aspects related with figure storage and manipulation;
4. Show the main topics taught at geometry classes and why they are useful for CAD systems, including pattern matching of geometric elements (Hough transforms) and image scaling, dislocation and rotation;
5. Show what is computer vision and how geometric elements can be used in the determination of specific shapes, as circles, squares and lines, or, in real life, apples in a tree, cookies in a bakery, hole of a car's fuel reservoir in a gas station, or a stick in the middle of the road, for example.

During the lectures on this subject the instructor must emphasize that CAD is a geometrical tool. The depth of the geometry presented in step four is quite intuitive. At this moment the students were just introduced to analytical geometry (we are just three to four weeks from the semester's start) and, therefore, should not be overloaded with in-depth concepts. Hough transforms are cited, and superficially described as a tool that given a set of points try to figure out if a specific geometric element contains such a set or one of its subsets, in order to give a tool exclusively based on geometric functions.

These applications are very appealing since the students can actually see what is being told to them. Although both applications are bended to imagery we could use different perspectives of images, such as industrial design, auto-guided vehicles (including our cars in a near future), artificial vision for humans, etc. At this time these applications are very well succeeded in achieving the planned goals, as shown in the final section.

4.2 Introducing calculus

This course's section uses a different approach to show the importance of calculus for computer science. Instead of showing an application that uses calculus explicitly, we use an application that uses another mathematical tool that uses calculus directly. This inversion is intentional and is performed in order to make the students understand that a basic course is a basic course and could never be used directly into their professional lives but could be the basis for a different course that has that direct application. The starting application here is image processing (using statistics) and the steps to be followed are:

1. Show how digital images can be enhanced through filtering, that is, use Khoros[1] to experiment few filters, specially histogram equalization;
2. Show some applications of histogram equalization as a filter, using some images prepared to cause impact and other to cause frustration;
3. Describe what means histogram equalization in terms of the graphic areas, showing how the redistribution of these areas cause the image improvements or distortions;
4. Define how continuous histograms are equalized by an integration operator, leading to the need of integrals;
5. Show informally the notions of integrals and probabilities;
6. Show what performance analysis is, what are its applications, and how it can be achieved through probabilistic models (markov chains, queueing theory).

This section is particularly challenging since we start from a theory much ahead of the students knowledge, forcing them to realize that limits and derivatives are just a starting point for integrals and that these are also the starting point for probabilities, which have a wide range of applications inside computer science. During the last step of this section the students are encouraged to write a small program that generates random numbers using uniform, normal and exponential distributions. This exercise shows them the difference in the models used to analyze the performance of a computational system.

5 Conclusions

The results provided by the introduction of the course just described into our curriculum were very stimulating. A short summary of these results is provided at Table ??, where we include data about the abandoning rate (during the freshman year) on the last five years and the rate of students' failures in the math courses. The results on this table integrate only data about freshman students and the rates of failure are the average between four distinct courses of mathematics (Calculus I and II, Vectors and Analytical Geometry and Linear Algebra). All freshmen are included in the study and the courses were restricted to those scheduled to the curriculum's first year in order to provide a common base for all students since in 1997 we had a curriculum change that included different math courses.

[1] Khoros is an image processing tool created by Khoral Research Inc.

Table 2: Abandoning and failure rates.

Year	Abandoning (%)	Failure (%)
1994	23.3	18.3
1995	20.0	18.2
1996	18.2	15.2
1997	9.1	6.7
1998	6.1	8.1

As one can see, the two classes in which this course was official (following the new curriculum) had results during the freshman year much better than the previous ones. The pilot experience in 1996 only produced marks slightly better than those achieved in the previous years. This fact is admissible because that was our first experiment and many mistakes made there could be fixed during the following years.

What actually should be considered is the decrease in the number of abandons in the last two classes, which dropped to one third of the old rate. For university plannings and organization rates under 10% (as they are now) are highly desirable and do not imply in extra care from the faculty body. In our case the old rates were really troublesome since this is a government funded university, which works exclusively from tax payers funding. Therefore, each abandon implied in a bad tax use and must be avoided.

The rate of failures is now on an excellent level. The impact of these results in the quality of the professional graduated in such conditions cannot be exactly measured at this time. However, the performance of students that went through this approach in the advanced professional courses has been consistently higher than other students, what may be a strong indicator that their quality is also improving.

From these remarks we may conclude that this approach provides very interesting results in the task of turning math attractive and reducing the abandoning rate. Since it does not request extra resources from the college and does not need a specific preparation for the instructor, except the need for versatile instructors, we decided to keep it as an introductory course and also encourage its application at other universities, what may expand the methodology and the range of examples given in the classrooms.

References

[1] ACM/IEEE-CS, "ACM Curricula Recommendations Volume I: Computing Curricula 1991", ACM Press, 1991. (available at *www.acm.org/education/curricula.html*)

[2] Cordes, D. et al, "An Integrated First-Year Curriculum for Computer Science and Computer Engineering", *Proceedings of the 1997 Frontiers in Education Conference*, November 5-8, 1997, Pittsburgh, PA.

[3] Richardson, J. et al, "Freshman Design Projects in the Foundation Coalition", *Proceedings of the 1998 Frontiers in Education Conference*, November 4-7, 1998, Tempe, Arizona.

[4] Sociedade Brasileira de Computação, "Currículo de Referência 1996", available at *www.sbc.org.br/cr/crf96.html* (in portuguese), 1996.