IETC 2014

Increasing practical lessons and inclusion of applied examples to motivate university students during programming courses

A. Ramirez-Lopez*\(^a\), D. F. Muñoz\(^a\).

\(^a\)Department of Industrial & Operations Engineering, Instituto Tecnológico Autónomo de México. Rio Hondo # 1, 01080 México City, Mexico.

Abstract

This study investigated the influence of increasing practice and the inclusion of applied examples (exercises with a direct application or related to real problems) during programming and numerical analysis courses. The results obtained from this experiment demonstrated that examples related to students engineering careers have a very positive impact on learning and awakened the student interest on programming courses; in comparison with students who just follow examples from a textbook as in an ordinary course. The conclusions of this work were obtained from the analysis and comparison of the statistical information of the students' performance during 2 programming courses and an additional mathematical course (numerical analysis). Finally, a questionnaire (at the end of the courses) about the methods used for teaching was applied in order to investigate students' opinion.

© 2015 The Authors. Published by Elsevier Ltd.

Peer-review under responsibility of the Sakarya University.

Keywords: Increasing practical lessons; Teaching/learning strategies; Real-life examples; Programming courses.

1. Introduction.

Programming and computational methods have been developed to create very friendly and interactive virtual environments for all users of digital services like computers and cell phones, internet and TVs programs. Nevertheless, people often forget how these gadgets and software were created and just enjoy about; people usually do not think about the work behind. Maybe this is the reason why programming and computational courses, (including some mathematical courses like numerical analysis) are frequently considered only as complementary courses during the university student formation. Students who take these courses (including students in sciences and engineering) frequently ask themselves if it is required to take them or if they have any application on their

* Corresponding author. E-mail address: adan.ramirez@itam.mx
corresponding career. They frequently ask themselves why I have to learn to program in any programming language. May be there is a software already developed to do it in somewhere or it is possible to download an app; thus they decline importance to these courses. They think that it is more important to pay attention on other courses, for example, mechanics, aerodynamics, chemistry, engines, thermodynamics, technology, etc.

Some authors have been working on the development of new and more efficient techniques and methods to improve learning using new technologies or creating didactical software (Depradine et al. 2004, Smith et al. 2006, Ellis et al. 2006 & Doornekamp et al. 1993), some of them developed support learning materials and implemented evaluation systems for students and the methods used to teach and learn, some others have studied the student behavior towards computers (Donker et al. 2007, Lang et al. 1992 & song et al. 1997), others have studied the ability of the students in different school levels to use computer equipment and accessories like keyboards, mouse etc. and assessed their response to the use new technologies in the classroom, including the use of graphical and interactive tools in order to help the students to understand some problematic courses like physics, mathematics or chemistry, where the use of audio-visual material is widely recommended for students on high schools and universities (Donker et al. 2007 & Henning et al. 1993). All these studies have been providing important information about the way to improve the teaching and learning processes using new technologies. Nowadays the contact with computers, virtual environments & electronic devices has been increased since the students were children. Undergraduate and graduate students were born at the end of 80’s or at the beginning of 90’s. They are between 17 to 26 years old. They belong to a new generation. Nevertheless many of them have trouble understanding programming courses and others related to mathematical, chemical and physics topics because the notation and the concepts are abstracts and new (Du Plessis et al. 1995, Rowe et al. 1999 and Cox et al 1994). Thus this work was focused on investigating the influence of new methodologies that includes the solution of some real-life problems and increasing practice during programming lessons in order to motivate the students for learning about programming and feasible applications (Hietala et al. 1993).

2. Programming and numerical courses.
Two samples of engineering students for each of three courses were considered to participate in this study. They were enrolled in the following courses:

1. Programming course I
2. Programming course II
3. Numerical analysis

The contents of the courses are described next:

A first programming course is elementary to get the students feet wet, they learn the basics of a programming language and are introduced to the integrated development environment (IDE) of the corresponding programming language. Then, they continue learning how to define data types and they learn about the language basic capabilities and management. Subsequently, the students are involved on the language syntax, they study the difference between expressions and statements and the use of punctuators and operators; so that students learn about the use of functions, classification and work modes, their importance and applications. At the end of the course the students make their own programs using advanced concepts like arrays management, loops nested commands and routines. Students learn to employ instructions and create their own routines and subroutines to be included in a main function. Finally the students develop their own header files too. Although the programs developed during this course are very easy to program, not all of the students finish in satisfactory terms.

Nowadays university students are strongly influenced by video games and Internet that they think that this kind of programs are older, ambiguous and bored, and their interest on programming is frequently missed. This is the reason why motivation is a very important factor for a better student performance.

In the second course (programming course II), students receive information about hot new rapid application development (RAD). These applications are used in visual programming languages that allow them to take advantage of object-oriented programming (OOP). Students can write Windows programs more quickly and more
easily than ever possible before thanks to the new programming languages. They can create complex applications with Win32 GUI (graphical user interface) programs. In other words they can create the user interfaces to a program (the user interface means the menus, dialog boxes, bottoms, pop ups, windows, etc.) and so on using drag-and drop techniques for true rapid application development. They can also drop controls on forms to create specialized programs such as Web browsers in a matter of minutes. After this, they have the ability for modifying the component properties and events of their programs. At the end of the course students learn about the basic instructions to initialized graphical process on the screen.

Numerical analysis is the third course; this is especially difficult for all the students, because it is necessary to have a good knowledge of mathematical and programming concepts. A very good understanding of loops and nested loops is desirable in students because these are platforms to develop the iterative routines to find mathematical solutions with numerical methods. The use of graphics is also recommended for a better understanding, for this reason, the authors have included the use of some mathematical software such as Excel, Mathematica, MATLAB, MAPLE, etc. in addition to the traditional routines and flow charts developed for programming languages in order to show a new way to learn these themes. The numerical analysis course is divided in 4 stages during which the students must learn to implement different numerical methods as is shown in Table 1. Every stage is focused on teaching a different mathematical aspect of numerical analysis.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Theme</th>
<th>Particular topics</th>
</tr>
</thead>
</table>
| 1     | Solving Nonlinear Equations     | Newton’s method.
|       |                                 | Method of false position.
|       |                                 | Muller’s method.                                                                |
| 2     | Solving set of equations        | Matrix notation.
|       |                                 | The elimination method.
|       |                                 | Gauss and Gauss-Jordan methods.
|       |                                 | Determinants and matrix inversion.                                              |
| 3     | Interpolation and curve fitting | Lagrangian polynomials.
|       |                                 | Interpolation with a cubic spline.
|       |                                 | Bezier curves & B spline curves.
|       |                                 | Euler & Heun methods.                                                           |
| 4     | Numerical Differentiation & Integration | Extrapolation techniques.
|       |                                 | Simpson’s rule.
|       |                                 | Trapezoidal rule.
|       |                                 | Gaussian quadrature.                                                            |

3. Participants (samples and students).
Six groups of undergraduate (engineering) students were considered in the present work. They were classified in 2 samples (A and B) for each of the 3 previously mentioned courses. Their designation and corresponding populations are shown in Table 2. Students in any course were randomly assigned to group A or group B. Their school records, previous formation, social status are also variables, and these factors were not considered in this work. Nevertheless some of them had a previous experience on programming due to their formation as computer technicians that help them to obtain a better performance than students who did not have. The population of the original groups was reduced for numerical analysis courses because some students did not approve the programming courses and, in consequence, they were not authorized to take the last course. All of the participants received the same courses with the same containing but some differences were implemented during teaching process to the students of samples as is described next.

4. Teaching methodologies.
Professor for sample “A” only followed the textbook examples and no additional information was provided to the students. All the topics were treated as traditionally. Nevertheless, a good way to motivate the students was to increase the contact with them during practical lessons for groups of sample “B”. A brief explanation of the topics was taught in theoretical lessons and examples related to their career (aeronautical engineering) were included. The programs and exercises solved by students were developed profiting the professor’s personal and professional experience.
Programming courses I and II are taught weekly in 4 lessons of 90 minutes each. Students of sample “A” had 2 theoretical and 2 practical lessons, but students of sample “B” had 1 theoretical and 3 practical lesson. Numerical analysis course is taught in a similar way in 4 lessons weekly of 90 minutes although due to the complexity of the mathematical topics 2 lesson are dedicated to theoretical topics; here the students learn the nature of the methods and solve some cases using only calculators pen and paper. The other 2 lessons are practical; here the students run programs for the corresponding numerical method.

5. Evaluation of the student performance.

Student assessment is the result of 4 partial evaluations for the programming and the numerical analysis courses every one is 25% of the final assessment. The first 3 of them are partial evaluations referred to the stages of the course. Every partial evaluation consists of a pair of tests, one is theoretical and the other is a practical test. The same test was applied to all the students in both samples; and for the final stage, the students developed a final project; then they had to work in groups of 4 students and create a computer program. The students of the samples were randomly divided forming teams according with the group populations in order to develop their project. In the first course the projects were very simple but for the second course students of sample “B” decided to create a personal version of software to sell and print tickets for a virtual airline, including options to select different airplanes and different departures and arrivals, some teams used national destinations and others employed foreigner destinations.

During the development of the final project for the programming II course, students employed their imagination and knowledge to create useful software. They created an environment where the user can select a flight with a departure and a destination. Every team designed different airplanes sketches (some of them were Boeing 727, 747, 757 & DC10) using graphical instructions to be displayed on the screen. Registration of the passengers and seat assignment were done by using arrays and then stored and saved in new files. In addition, routines to change and correct information for passengers and reprogramming flights were included. Although the programs could not be placed on a website or generate a database, they were good examples for students who had to develop their own routines without downloading Internet files. While for the final project of the numerical analysis course students of sample “B” decided to develop a computational display for calculating the position of an airplane. This project involved the creation of reading data routines to define the initial position and the new conditions of the airplane for every attitude change. The airplane was treated as a simple particle; so that the movement was obtained from a simple integration method as a function of a step time (Δt) that controls the finest of the graphical display and the precision during the simulation; and some aeronautical quantities were calculated like the True air speed, the real air speed and vertical speed. Graphical routines to calculated and display the airplane updated position (displacement on the air & the on ground) were also programmed. The main screens for the final projects that were developed by students of the sample “B” are shown in Fig. 1 and Fig. 2, respectively. These programs are not complex or sophisticated as a professional simulator, but they are interesting examples of possible situations for the students to apply the knowledge learned.

In contrast, students of sample “A” were free to develop the project whatever they wished. Unfortunately, they had many problems. They did not know how they could use their programming techniques. It was difficult for them to find an application to their knowledge. Another disadvantage was their shorter practical time. These factors limited the examples and exercises treated in class. In consequence the quality of the projects presented by students of sample “A” was considerably inferior.
6. Student Assessments, Analysis and Results.

Some aspects like student attendance, and their performance during the courses were taken as parameters to evaluate the effectiveness of the methods used to teach. The number of practical lessons was increased for student of sample “B” they had a (75%) of the course time in practical lessons; in contrast students of sample “A” only received (50%) of the total lessons working on programming practices. Fig. 3 shows a comparison between the class assessment obtained by the students of the 6 groups analyzed. It is clear that students of sample “B” got the best assessment during all the courses. While Fig. 4 shows the average of the class assessment obtained by every group of the sample “B” during the 3 courses. The final assessment scale goes from zero to ten. Six is the minimum required to approve a course. Statistics of the total approved, non-approved students and their corresponding average assessment during the 3 courses are shown in Fig. 5 and tables 3 to 6. As can be noticed, the average of class assessments for students of sample “B” was considerable superior than those of sample “A” in every participant group. This difference is increased if the students who did not approve the course are considered. In addition, the percentage of non-approved students in sample “B” was reduced considerably in comparison to students of sample “A”.

---

**Fig. 1.** Main screen of the project developed by student of sample “B” at the end of programming course II.

**Fig. 2.** Main screen of the project developed by student of sample “B” at the end of Numerical analysis course.
a) During programming courses.

![Graph showing class assessment records between students of samples “A” & “B”.

b) During numerical analysis course.

![Graph showing class assessment records for the groups of sample “B” during programming courses.

Fig. 3. Comparison of class assessment records between students of samples “A” & “B”.

Fig. 4. Class assessment records of the groups for sample “B” during programming courses.

Table 3. Class assessment for sample “A” (total average) during programming courses.

<table>
<thead>
<tr>
<th>Groups</th>
<th>1A1M</th>
<th>1A2MV</th>
<th>1A3M</th>
<th>Final average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class assessment Averaged</td>
<td>5.1152</td>
<td>4.7654</td>
<td>4.7069</td>
<td>4.8625</td>
</tr>
<tr>
<td>Class assessment averaged considering Only approved students</td>
<td>7.9187</td>
<td>7.5845</td>
<td>7.7726</td>
<td>7.7586</td>
</tr>
</tbody>
</table>

Table 4. Class assessment for sample “B” (total average) during programming courses.

<table>
<thead>
<tr>
<th>Groups</th>
<th>1A4V</th>
<th>1A5V</th>
<th>1A6V</th>
<th>Final average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class assessment Averaged</td>
<td>7.5862</td>
<td>8.3333</td>
<td>7.6667</td>
<td>7.8620</td>
</tr>
<tr>
<td>Class assessment averaged considering Only approved students</td>
<td>7.8571</td>
<td>9.2592</td>
<td>8.2800</td>
<td>8.4654</td>
</tr>
</tbody>
</table>
Table 5. Performance of the students of samples “A” & “B”.

<table>
<thead>
<tr>
<th>Course</th>
<th>Programming I &amp; II</th>
<th>Numerical Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>Approved</td>
<td>36.25</td>
<td>91.86</td>
</tr>
<tr>
<td>No approved</td>
<td>63.75</td>
<td>8.14</td>
</tr>
</tbody>
</table>

Table 6. Performance of the students of samples “A” & “B” with or without a previous programming formation.

<table>
<thead>
<tr>
<th>Approved</th>
<th>Group</th>
<th>1A1M</th>
<th>1A2M</th>
<th>1A3M</th>
<th>1A4V</th>
<th>1A5V</th>
<th>1A6V</th>
</tr>
</thead>
<tbody>
<tr>
<td>With</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Without</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>15</td>
<td>18</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>11</td>
<td>9</td>
<td>9</td>
<td>25</td>
<td>27</td>
<td>27</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No approved</th>
<th>Group</th>
<th>1A1M</th>
<th>1A2M</th>
<th>1A3M</th>
<th>1A4V</th>
<th>1A5V</th>
<th>1A6V</th>
</tr>
</thead>
<tbody>
<tr>
<td>With</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Without</td>
<td>18</td>
<td>16</td>
<td>15</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>19</td>
<td>16</td>
<td>16</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

a) during programming courses

b) during numerical analysis course

Fig. 5. Comparison about the performance of the student of samples “A” & “B”.

Numerical analysis is considered the most difficult course, the performance and class assessment was lower than for programming courses, as is shown in Fig. 3 and Table 5, including for students of sample “B”. Nevertheless the
performance of these students remained better than students of sample “A”. The percentage of unauthorized students to take this course was 21.25% (of the original population) for sample “A” in contrast to only 6.98% of the students for sample “B”.

In addition, the students’ class attendance was considered as another good indicator to evaluate the students’ interest on the course. Fig. 6 shows the statistics from classroom attendance for students of sample “A”. It can be noticed that attendance remains almost constant during the courses. In Fig. 7, a comparison between the 2 samples is shown. Attendance of the students in the 2 samples remains nearly constant at the beginning of the courses. Nevertheless it was observed for the 3 courses that after the second evaluation, students of sample “A” had a lower performance. Unfortunately some of them missed the possibility to approve the courses and therefore attendance was considerably lowered during the last stages, and the final average assessment of these groups went down. Many students missed interest in the course. In contrast students of sample “B” had a better performance in practice and theoretical tests. The student presence remained (80-90%) until the final due to they remained motivated about the topics of the course because they had the opportunity to develop a project related with their career.

![Fig. 6. Presence records for students of the sample “B” during programming courses.](image)

**Fig. 6.** Presence records for students of the sample “B” during programming courses.

(a) During programming courses.

(b) During numerical analysis course

![Fig. 7. Comparison of presence records between students of the samples “A” & “B”.](image)

**Fig. 7.** Comparison of presence records between students of the samples “A” & “B”.

7. **Students Opinion.**
The students who participated during programming courses I and II were classified in 2 classes as it was shown in Table 2, some of them with and some others without a previous programming formation or experience in computers and programming. Students who had received a previous formation as a programming technicians in High School found the course interesting, they commented that the tests applied were easier to solve than they expected; in contrast, students without a programming formation found more difficult to understand the programming courses, nevertheless at the end of the courses, students of the sample “B” commented that programming was not very difficult at all. These students concluded that they had not imagined the possibilities of programming development with an aeronautical application. Finally they told, felt happy to have found a connection between the course and their university studies.

It is important to illustrate that, for the sample “A”, 83% of the total approved students were students with a previous background in programming, only 17% of the students without a previous programming background approved the courses; unfortunately 4% of the students who had a previous background did not approve the courses. In contrast, all the students with a previous background approved the courses of sample “B”. The percentage of students without a previous background who approved the courses was also increased as is shown in Table 7 and Fig. 8.

<table>
<thead>
<tr>
<th>Professor</th>
<th>Method for instruction</th>
<th>Group</th>
<th>Original groups</th>
<th>Students for numerical analysis course</th>
<th>Students with a previous programming formation</th>
<th>Students without a previous programming formation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Traditional</td>
<td>1A1M</td>
<td>30</td>
<td>22</td>
<td>12</td>
<td>18</td>
</tr>
<tr>
<td>A</td>
<td>New</td>
<td>1A2M</td>
<td>28</td>
<td>26</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>B</td>
<td>Traditional</td>
<td>1A3V</td>
<td>27</td>
<td>20</td>
<td>10</td>
<td>17</td>
</tr>
<tr>
<td>B</td>
<td>new</td>
<td>1A4V</td>
<td>30</td>
<td>27</td>
<td>12</td>
<td>18</td>
</tr>
</tbody>
</table>

Fig. 8. Compassion between approved and non approved students during programming courses as a function of their previous formation (with or without previous computational studies).
During the numerical analysis course, nearly 70% of the students had troubles understanding the topics of the course, including students of both samples and students with or without a previous programming background.

Mathematical courses are always difficult for the students; for numerical analysis it is necessary to know the basic concepts of geometry and calculus. Numerical methods request to understand and repeat a procedure in order to obtain the solution.

The students answered the following questionnaire after they concluded the 3 courses. The questions were designed to know the students’ opinion about the teaching methods used by professors; a little modification on question “9” was required because of the differences in the methods used.

The results obtained from the questionnaire are shown in the graphics of Fig. 9. Students of sample “B” considered that the examples used by the professor in lessons were more appropriate than those used by the professor of sample “A”, many students of sample “B” considers that they really learned about programming and numerical analysis. They think that these courses really can help to solve problems in the future or in a real workplace.

During lessons the students of sample “B” considered as good the communication with the professor and many of them said that they had paid attention and feel motivated. In the other hand students of sample “A” considered that they felt frustrated and unhappy due to the absence of a relation with the rest of knowledge acquired.

The result of the question (9) is shown separately in Fig. 10 to evidence the student feelings. They think it was a good idea to increase practice time during programming and mathematical courses for a better understanding.

Difficulties in terminology, definitions, and introduction to topics related to computer science are examples of frequent problems, which difficult the students’ understanding of programming topics during the courses. Some authors have studied this situation (Asai et al. 1991, Woodhouse et al. 1983, & Brusilovsky et al. 1997) and have developed some methods and techniques to improve the way to introduce the students to programming. An example is the mini-languages approach. The main idea is to design a small simple language to support the first steps in learning programming. Many of the mini-languages have been developed as tutorials and use an actor to guide the users, they are considered as a powerful tool to introduce students to programming. Tutorials and interactive guides are good ways for a better learning during programming courses; these works develop the student intuition, (Baldwin et al. 1999, Rocchi et al. 1998, Dagdilelis et al 2001 & Milne et al. 2002). The promise of multimedia learning is that students can learn more deeply from well-designed multimedia messages consisting of words and pictures than from more traditional modes of communication (Van der Meij et al 2006, Schmitz et al. 2006 & Summers et al. 2007). Nevertheless the application of programming in sciences is wide; thus it is required to develop particular tutorials for students of every career. A good option is to profit the professional experience of the professors; and begin to create personalized exercises, tutorials.

<table>
<thead>
<tr>
<th>Question</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.- Were appropriated for the courses the examples employed by the professor?</td>
<td>a) yes b) sometimes c) no</td>
</tr>
<tr>
<td>2.- How do you consider your courses?</td>
<td>a) interesting b) bored c) without purpose</td>
</tr>
<tr>
<td>3.- do you think you learn programming in the courses</td>
<td>a) yes b) enough c) a little bit</td>
</tr>
<tr>
<td>4.- The project developed really helped you to apply your knowledge?</td>
<td>a) yes b) partially c) no</td>
</tr>
<tr>
<td>5.- Do you think these course will help you to solve problems in the future?</td>
<td>a) yes b) maybe c) never</td>
</tr>
<tr>
<td>6.- How do you consider it was the communication (student-teacher)?</td>
<td>a) good c) regular d) bad</td>
</tr>
</tbody>
</table>
7.- How did you feel in the classroom?  
a) motivated  
b) bored  
c) uncomfortable  

8.- Did the professor help the students if they were in problems or during lessons?  
a) yes  
b) sometimes  
c) no  

9.- Do you think it cold be good to increase the practice time and reduce theoretical lessons and include examples with direct or real application on the course?  
a) yes  
b) no  

9.- Do you think it was good to increase the practice time and reduce theoretical lessons?  
a) yes  
b) no  

Additional comments:
8. Conclusions & comments.

It is possible to make the following conclusions & comments about this work.

New student generations demand new methods to teach and learn.

Motivation is one of the most important factors that influence the effectiveness of explanations and understanding on students, especially in classes where the topics are abstract or difficult to explain like mathematics and programming.

Increasing the practical lessons was a good idea to avoid students were bored or upset on the classroom.

Examples related with real problems and situations or with direct applications are learned easily by students.

Final projects helped the students to find a direct application of their new knowledge. They discover that they are capable to create interactive tools for solving problems related with their careers.

Programming is an opportunity for the students to develop their abilities to treat a problem.

Increasing of practical lessons helps the students to develop abilities and criterion.

Professors with teaching experience and professors with industrial experience must work together in order to develop more interesting examples and better methods for teaching.

Acknowledgments
The authors wish to express their gratitude to the Instituto Tecnológico Autónomo de México, Consejo Nacional de Ciencia y Tecnología, and Asociación Mexicana de Cultura A. C. for their valuable support.

References


Iain Milne & Glenn Rowe, Difficulties in Learning and Teaching Programming—Views of Students and Tutors, *Education and Information Technologies*, vol. 7 Issue 1 (2002), 55–66.

Jan van der Meij & Ton de Jong, Supporting students’ learning with multiple representations in a dynamic simulation-based learning environment, *Learning and Instruction* vol.16 Issue 3 (2006), 199-212.
