Bloom's Taxonomy Based Approach to Learn Basic Programming Loops

Anabela Gomes
Informatics Engineering Department
Coimbra Polytechnic - ISEC
Centre for Informatics and Systems
University of Coimbra
Coimbra, Portugal
anabela@isec.pt

Fernanda Brito Correia
Informatics Engineering Department
Coimbra Polytechnic - ISEC
Coimbra, Portugal
Institute of Electronics and Informatics Engineering of Aveiro
University of Aveiro
Aveiro, Portugal
fernanda@isec.pt

***Abstract*— This Research to Practice Full Paper presents a study on the difficulties 46 Biomedical Engineering students show in a particular topic of the first programming course, the loop topic. Students' results in programming courses are often very disappointing and to improve these results it is fundamental to understand the type of difficulties students feel when they are leaning how to program. Most of the researches are too generic and do not report on specific programming topics and we consider important to be more specific. To achieve this purpose we organized a test, using the revised Bloom's Taxonomy, where all of the questions were about the programming loop topic. We did a statistical analysis based on the grades students obtained on each question of the test and we concluded that the approach we followed has clear pedagogical advantages to weaker students, making them more confident and motivated and allows teachers to use adequate strategies to each of the identified students' difficulties. The authors propose, in the next future, a set of visual methodologies to be used with the students, which they expect can contribute to delineate a more motivating teaching strategy in overcoming students' difficulties.**

Keywords — Bloom's Taxonomy, Computer Science Education, Programming Teaching Strategies

I. INTRODUCTION

Learning computer programming is a difficult process for many students worldwide. The literature includes many descriptions of failure cases all over the world, independent of programming languages or paradigms used [1]. Some authors have discussed different reasons for such problems [2-6]. We also think that there are several reasons that cause these difficulties and could be pointed out, like teaching methods, study methods, student's abilities and attitudes, the nature of programming, the type of students' motivation and also the strategies used by the teachers [7]. In previous researches we investigated the impact that some personal characteristics may have had in the performance of learning to program [8, 9]. However, previous programming experience showed an important factor influencing the results in learning how to program, even when using a different language or paradigm. This is not surprising, as we believe that general programming abilities are far more important than language specific details and that it takes time to train and mature the acquired knowledge.

Many solutions have been proposed in the literature and, although many of them may give very important contributes to the resolution of this issue, in general, the problem remains [10-19]. We believe, that to modify the current picture in introductory programming courses it is necessary to improve several areas, such as, student motivation, attitudes and study and teaching methodologies. However, first of all, it is fundamental to understand the type of students programming difficulties. Some observations and studies about the main errors made by students when they start programing, whether during the class, the exams or other types of environments were already made [20]. Different approaches and tools have been proposed to help students and teachers reach their objectives, but most of the researches are too generic and do not report on specific programming topics. We consider it is important to be more specific and report about specific programming topics, through more deep studies and analyzes of the themes in which the students have more difficulties and of the way to surpass them and propose specific pedagogical activities for each of them.

Many times, programming teachers ask students to develop complete programs beginning in the early stages of the course. Normally, simple programs are asked in early stages, and problem complexity grows as the course develops. We believe that this strategy may be inadequate for many students, especially those that can't create programming solutions even to simple problems. That means that the complexity is too high for these students right from the beginning, leading to frustration, lack of motivation and failure. At the same time, this approach, makes it difficult for teachers to clearly identify the type of students' difficulties. As mentioned in [21], "students should first be taught to read programs before they write programs. As children, in our early school years, the ability to read and understand our native language outstripped our capacity to write fragments of that language".

In this paper, we describe a study done on the introductory programming course of the 1st year and 1st semester of Biomedical Engineering degree of our institution, the Informatics and Systems Department (DEIS) from the Engineering Institute of Coimbra (ISEC) of the Polytechnic Institute of Coimbra (IPC). This course uses as introductory language the C language and the subjects covered include the basic concepts taught in an introductory programming course

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

using a procedural programming language, like data types, operators and expressions, standard input and output formatted data, data structures, loops, functions, arrays and string manipulation. In the first three weeks, the students solve programming problems using sequential, selection and repetitive structures through pseudocode. Only after a certain comprehension of the subject the problems are solved using the C language giving emphasis to the syntax details. This course has 4 contact hours per week, 1 hour of theoretic class and 3 hours of lab classes in two groups of about 30 students each. In addition, teachers offer more 6 hours per week to clarify students' doubts, during the entire semester.

The observed problems, collected during the classes and also by analyzing the tests and exams, indicated to us that the great difficulties begin with loops. Hence, in this study, we concentrated on analyzing this topic. A taxonomy of educational objectives, such as revised Bloom's Taxonomy of Educational Objectives, can be a good reference to understand this question, since it is divided in levels that can be translated to an assessment of increasing complexity level. We think that before students write complete programs they should master easier tasks that can be related with the first and intermediate levels of the Bloom's Taxonomy.

We agreed that the different activities and problems proposed to students during the classes should follow an increasing level of difficulty, but the entry level should be easy enough, so that most students can have success in the early learning activities and feel motivated, so we organized a test about the programming topic looping (all of the questions were about this topic) in terms of the Bloom's Taxonomy followed by a statistical analyses of the results to make conclusions about the pedagogical advantages of this approach.

II. TAXONOMIES OF LEARNING

Taxonomies of educational objectives are used worldwide to describe learning outcomes and the assessment results, reflecting a student learning stage. Usually they divide educational objectives into three domains: cognitive, affective and psychomotor. As described in [22], some, such as Bloom's taxonomy, treat each of these as a one-dimensional continuum, others, like the revised Bloom's taxonomy describe the cognitive domain using a matrix. Yet others, like the SOLO taxonomy [23], use a set of categories that describe a mixture of quantitative and qualitative differences between student performances. There are also taxonomies that use all three domains equally. Existing researches on the use of learning taxonomies in computer science focuses mainly on the cognitive domain and among the diverse taxonomies proposed in the literature, a very well-known taxonomy is the original Bloom's taxonomy [24]. It is a classification of the different objectives and skills that educators set for students, where the skills in the cognitive domain are divided in six levels. Starting from the lowest order processes to the highest, those levels are: "Knowledge", "Comprehension", "Application", "Analysis", "Synthesis" and "Evaluation". Higher levels are built on lower ones and are considered more complex and closer to complete mastery of a subject matter.

Bloom's taxonomy has been revised [25]. The authors changed the nouns listed in Bloom's model into verbs, to correspond to the way learning objectives are typically described (see Table 1).

TABLE 1. REVISED BLOOM'S TAXONOMY

Categories	Cognitive Processes
"Remember"	Recognizing, Recalling
"Understand"	Interpreting, Exemplifying, Classifying, Summarizing, Inferring, Comparing, Explaining
"Apply"	Executing, Implementing
"Analyse"	Differentiating, Organizing, Attributing
"Evaluate"	Checking, Critiquing
"Create"	Generating, Planning, Producing

Fuller and colleagues made a revision about existing taxonomies, stressing the strengths and weaknesses of each one. They also proposed a new taxonomy and discussed how this can be used in application-oriented courses such as programming [21].

Although we find several positive aspects in the approaches followed in different taxonomies, we agree with Raymond Lister when he says that "specific to the teaching of elementary programming: the lower two levels should emphasize the skill of reading and comprehending code, the intermediate two levels should emphasize the writing of small fragments of code, but within a well-defined context, and the upper two levels should emphasize the writing of complete non-trivial programs" and that "students should first be taught to read programs before they write programs" [21]. We chose to develop this study based on the revised Bloom's Taxonomy.

III. THE STUDY

A total of 46 students (16 male and 30 female) appeared for the test we structured based on the Bloom's taxonomy. The number of freshmen students was 36 (11 male and 25 female). For the purpose of our study we considered only freshmen, as we wanted to consider only students who were in the same circumstances when the course started.

We structured the test (2nd test), according to the Bloom's taxonomy. That is, instead of asking students to write complete programs (as was usual before), we structured the test in the following 6 questions (Q1 to Q6) and, to analyse student performance, we considered any answer to each question with a positive grade if it had a grade superior or equal to 50%.

The statistical results, calculated in percentage for each of the questions, were the minimum value (Min), the maximum value (Max), the average (Average), the standard deviation (Std. Dev.) and the median (Median) and can be found in Table 2 and in the bar graph of the Fig. 1 can be visualized the respective average with standard deviation and the median values.

TABLE 2. STATISTIC RESULTS OF THE QUESTIONS 1 TO 6

Level	Min	Max	Average	Std. Dev.	Median
Q1 - Remember	0.0%	100.0%	64.3%	48.5%	100.0%
Q2 - Understand	0.0%	100.0%	66.8%	31.0%	75.0%
Q3 - Apply	5.0%	100.0%	42.9%	27.9%	40.0%
Q4 - Analyse	0.0%	100.0%	18.2%	39.0%	0.0%
Q5 - Evaluate	0.0%	100.0%	30.0%	29.0%	25.0%
Q6 - Create	0.0%	100.0%	28.7%	25.9%	22.5%

a) The first question (Q1) was a question belonging to the Knowledge level ("Remember"). This question consisted of a code statement for students to assess the need for a loop. This code statement asked the user his/her age that would be valid only when the value entered was positive. Although this is a very simple question, it refers to an error that occurs very frequently, when students cannot distinguish between a selection and a loop. Thus, the large amount of wrong answers was because students solved the question using a selection. Students forgot that a user may, more than once, enter an incorrect value (outside the expected range). The results obtained for this question were 0% and 100%, having fifteen 0% and twenty-seven 100%. Therefore, most students had satisfactory results in this question.

b) The second question (Q2), classified in the Comprehension level of Bloom ("Understand"), consisted of a multiple response question. The students should point out, circling the corresponding answer, which of the codes, printed in each line the double of each of the first "n" user-entered numbers, followed by a "*" on the next line. Options coded with various for loops and while loops were given to students and to minimize the chances of "guessing" by the students, wrong answers were penalized. The results obtained can be found in Table 2. Notice that only 8 students achieved 100%, having all code correct and not selecting any wrong option. Although the marks obtained in the answers to this question were high, we expected to obtain better results, since it corresponded to the basic understanding of the functioning of the loop.

c) The third question (Q3) belonged to the Application level ("Apply"), in which the students were asked to write a C program that receives an integer between 100 and 200 and presents the sum of its divisors. The student was also asked to make the program in a way requiring the user to enter a number in a specified range and to give an output similar to a given example. In this question, it was observed an abrupt decrease of the grades compared to the previous question. We think that this decrease is related to the fact that students have to write code. We want to highlight the fact that only 1 student obtained 100%, corresponding to a completely correct code. The difference in results obtained between the answers to this question and the previous question clearly reveals that there is a difference between the competence of undestanding code and the ability to produce code. Students who can interpret programs may not necessarily be able to write their own programs, being code interpretation and code writing two different aptitudes.

d) In the fourth question (Q4), at the Analysis level ("Analyse"), it was given to the student a set of lines of code that the student had to examine in order to be able to decide which output would be generated. This question, consisted in the creation of a visual pattern, triangle shaped, implying loops inside loops. For such, it was provided to the student, several similar outputs, but with differences to reveal the true understanding of the subject. Since only one response was correct, it was assigned 100% to the selection of the correct code and 0% to the wrong(s) answer(s). The decrease in the students' grades showed that only the best students were able to deal with the difficulty of this question. Notice that only 8 students achieved 100%, for having the correct code.

e) In question 5 (Q5), classified in the Evaluation level ("Evaluate"), the students had to assess code, justifying which of the 2 programs presented were more efficient to read 5 integer numbers and present the sum of positive numbers. The code, although had no chained loops, included a set of more elaborate code, considered by teachers as being more difficult to understand by students, like the inclusion of break and continue statements inside loops. Two very similar codes were provided, although only one was correct. There were few students that could do this identification and the level of justification was generally very poor, which revealed other weaknesses in reasoning and corresponding code translation.

f) At last (Q6), it was asked the student to create a program, much more complex, consisting in the generation of a graphic pattern that seemed a diamond, using a character requested to the user. This question, located at the highest level of Bloom's Taxonomy, the Creation level ("Create"), included a complex use of chained loops. To note that only 1 student got the answer completely correct, with 8 students with grade 0%. The average grade obtained to this question was quite negative.

Even though these questions were all easy, we noticed a decrease in the percentage of satisfactory answers from question 1 to question 6, which can be associated to a slight increase of difficulty and the need to produce more coding.

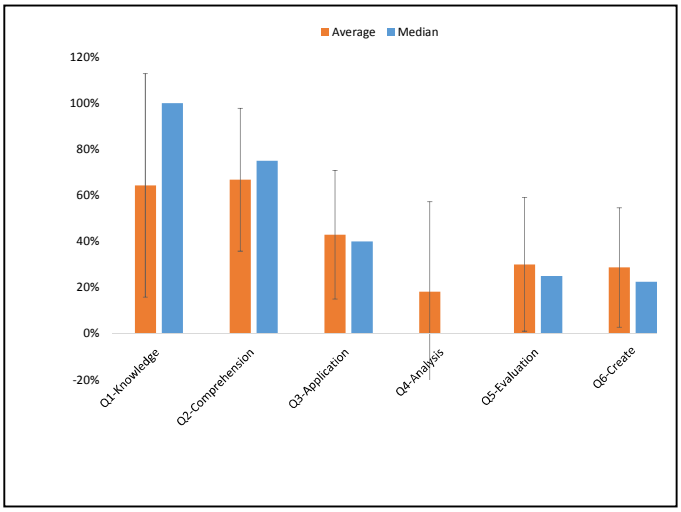

Fig. 1. Graph of the average, standard deviation and median values of the questions 1 to 6

IV. Discussion

The universal problems associated with programming education in many higher education institutions leading to high failure levels are well known. Our institution, is not an exception. From 2011/2012 academic years on, we have been teaching an introductory programming course and we have been more attentive to the type of students' difficulties and errors they make when programming. We have been gathering these types of problems and collected them during classes and exams. The classes' interactions with students enabled us to easily predict the success of students in introductory programming courses as well as the type of main topics where the principal problems happened. For our students the most problematic aspects are the chained loops and the functions. As chained loops are taught first, we decided to primarily concentrate on this topic. For that, we decided to make a test only about the loops topic, where each question could be classified as belonging to each level of the revised Bloom's Taxonomy. We wanted to find out in what Blooms level the students had more difficulties.

This is quite different from an approach where most activities ask students to develop complete programs that solve some proposed problem. Many students are not able to create those programs, which will install a feeling of inability, often leading to a lack of motivation and abandonment (typically programming courses have high abandon rates when compared with other courses). The approach we used, included low difficulty level activities, which gave weaker students a sense that they were able to do some activities correctly. This had some positive impact in the course. Many students commented that they were more motivated with this approach. They felt able to understand some programming and to answer correctly to some questions, thus giving them some sense of confidence. Even though the progress was slow paced, the students felt more confident and able to learn (some) programming. Students also commented that they become aware that learning to program is a process consisting of different stages and not a unique stage that is impossible to reach. They understood that it is a process that takes time and requires a lot of effort and intensive study methods in order to obtain positive results.

The results obtained, and their analysis showed us that even though it seems that students completely understand a loop when it is isolated, chained loops make the comprehension very difficult for them, causing students not to fully understand the interaction between the various variables in each of the loops, as well as all the interaction mechanisms between all of them. We also verified that the problem is much more complicated when, instead of using `while` loops, `for` loops are used. We think that the problem is due to the fact that the `for` loop can have a more compact visual representation, while in the `while` loop the visual representation is more elucidative of the dynamics involved.

We believe that not all students can become brilliant programmers, but we also agree that to be successful in a programming course, students should be able to build programs with some complexity. To reach this stage, takes time and, at each learning stage, teachers should take into consideration what the students know and don´t know. To the majority of students, learning should be progressive, starting with simpler questions (not complete programs), continuing gradually towards more difficult ones.

It should be noted that the proposed strategy is to be applied to the teaching of fundamental topics. We consider that students, to pass in an introductory programming course, must be able to develop complete programs and not only be able to read and interpret individual code fragments, so exams and other assessment methods should in the first place test this competence. However, we think that the proposed approach could be used in classes, intermediate formative tests and in some parts of exams, since it helps the identification of difficulties of a topic at various stages, while motivating students to learn more about that particular topic.

Finally, we consider that the approach we followed has clear pedagogical advantages to weaker students. They can become more confident and motivated. For teachers it is easier to identify each student's doubts and misconceptions and to help them to overcome those difficulties. Students' specific difficulties are more complicated to identify when they are asked to write complete programs due to the multiple concepts associated to this task.

V. Conclusions

It is known that programming learning is complex to many students. In the past we studied different factors that could contribute to this problem and several general causes were already identified. In this paper, we analyzed the specific types of students misunderstanding about loops, organizing a test with all question about this topic according to an increasing difficulty level following the revised Bloom's Taxonomy.

Although the final results were similar to those obtained in previous years, we noticed that the advantages of this methodology are twofold. In one hand, it caused an increase of motivation and confidence, especially among weaker students and in another hand, this approach facilitates the detection of students' difficulties by the teacher, which allow him to use strategies to attack each of the identified weaknesses. We believe that more time is necessary for weaker students to develop the necessary skills to be successful in the course as in fact, the final objective remains, being students to be able to develop programs to solve specific problems.

Our future plan is the application of the proposed methodology in another topic. Although we defend the traditional method to assess students, we agree that teachers can contribute to an improvement of the students learning. This can be done through the use of some taxonomy as reference for guiding the type of activities to be done in classroom, mainly to help teachers to detect students' difficulties. Teachers will be able to better diagnosis the students' difficulties and better guide them with suitable activities. The idea of this approach is also to define new pedagogical strategies, which, after being tested by students and teachers, can be followed in order to improve the process of programming teaching and learning.

Several authors had developed several types of tools to assist the teaching/learning of initial programming having in mind different approaches. However, it seems that these approaches don't contribute to some extent, to a full comprehension of all of

the programming topics. Therefore, we are planning a set of visual methodologies (static and dynamic) in order to test which ones leads to a better understanding of programming topics. We want to test them with students and we also want to have the contribution of a set of trained programming teachers. At the same time, we are not only interested to see which ones leads to a better understanding but also, which specific strategies contributes to a better engagement of the students with those particular topics. The idea would be to achieve a simultaneously more profitable and motivating strategy. This idea is currently being developed and the tests being planned.

We expect that our study can contribute to a valuable discussion about the impact of this approach in improving the students' learning process and the teaching methods used with them, by finding adequate strategies and types of activities that motivate the students in their process of learning. Additionally, we consider that it could be interesting to apply it in similar courses.

REFERENCES

[1] C. Watson, and F. W. Li, "Failure Rates in Introductory Programming Revisited," in *Proceedings of the Innovation and Technology in Computer Science Education Conference (ITiCSE2014), Uppsala, Sweden, ACM*, 2014.

[2] K. D. Sloane, and M. C. Linn, "Instructional Conditions in Pascal Programming Classes," in *Teaching and learning computer programming*, R. E. Mayer, Ed. Hillsdale, New Jersey, Lawrence Erlbaum Associates, 1988, pp.137-152.

[3] E. Soloway, and J. Spohrer, J. *Studying the Novice Programmer*. Hillsdale, New Jersey :Lawrence Erlbaum Associates, , 1989.

[4] T. Jenkins, "On the difficulty of learning to program," in *Proceedings of the 3rd Annual LTSN_ICS Conference (Loughborough University, United Kingdom, August 27-29, 2002), The Higher Education Academy*, 2002, pp. 53-58.

[5] E. Lahtinen, K. Ala-Mutka, and H-M Järvinen, H-M., "A study of difficulties of novice programmers," in *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (Monte de Caparica, Portugal, June 27-29, 2005), ACM New York, NY, USA*, 2005, pp. 14-18.

[6] W. D. Gray, N. C. Goldberg, and S. A. Byrnes, "Novices and programming: Merely a difficult subject (why?) or a means to mastering metacognitive skills*?," Journal of Educational Research on Computers*, vol. 9, no. 1, pp. 131-140, 2007.

[7] A. Gomes, and Mendes, A. J., "Learning to program – difficulties and solutions.," in *Proceedings of the International Conference on Engineering Education*, Coimbra, Portugal, 2007.

[8] A. Gomes, A. Santos, A., and A. J. Mendes, "A study on students' behaviors and attitudes towards learning to program," in *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education – ITiCSE2012 (Haifa, Israel, 2012), ACM Press, NY, USA*, doi:10.1145/2325296.2325331.

[9] A. Gomes, W. Ke,, M. J. Marcelino, S. Lm, S Siu, and A. J. Mendes, "A Teacher's view about introductory programming teaching and learning – Portuguese and Macanese perspectives," in *Proceedings of 45th ASEE/IEEE Frontiers in Education 2017 - FIE'17 (Indianápolis, Indiana, USA, 2017)*.

[10] R. Pattis, *Karel the Robot: A Gentle Introduction to the Art of Programming*. John Wiley & Sons, 1981.

[11] D. Arnow and O. Barshay, "WebToTeach: An Interactive Focused Programming Exercise System," in *Proceedings of the 29th ASEE/IEEE Frontiers in Education Conference,* San Juan, Puerto Rico, 1999, pp. 39-44.

[12] T. Naps, J. Eagan and L. Norton, "JHAVÉ – An Environment to Actively Engage Students in Web-based Algorithm Visualizations," in *Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education*, 2000, pp. 109-113.

[13] E. Roberts, "An overview of MiniJava," in *Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education (SIGCSE'01)*, Charlotte, USA, 2001.

[14] "Scratch" [Online). Available: http://scratch.mit.edu/. [Accessed January 2017].

[15] M. Ben-Ari, N. Myller, E. Sutinen, and J. Tarhio, "Perspectives on Program Animation with Jeliot," *Lecture Notes in Computer Science*, 2269, Springer-Verlag, pp. 31-45, 2002.

[16] M. Kolling, B. Quig, A. Patterson and J. Rosenberg, "The BlueJ system and its pedagogy," *Journal of Computing Science Education, Special Issue of Learning and Teaching Object Technology*, vol. 12, no. 4, pp. 249-268, 2003.

[17] T. Rajala, M.-J. Laakso, E. Kaila and T. Salakoski, "VILLE: a language-independent program visualization tool," in *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research* (Koli Calling '07), vol. 88. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 2007, pp. 151-159.

[18] A. Santos, A. Gomes and A. J. Mendes, "Integrating New Technologies and Existing Tools to Promote Programming Learning," *Algorithms*, vol. 3, no, 2, pp. 183-196, 2010.

[19] M. Ben-Ari, R. Bednarik, R., Levy, G., Ebel, A. Moreno, N. Myller, and E. Sutinen, "A decade of research and development on program animation: The Jeliot experience," *Journal of Visual Languages & Computing*, vol. 22, no. 5, pp. 375-384, 2011.

[20] A. Gomes, and F. B. Correia, "Errors to avoid in Programming Teaching and Learning," In *Proceedings of 7th International Conference of Education, Research and Innovation – ICERI'14* (Seville, Spain, 2014).

[21] R. Lister, "On Blooming First Year Programming, and its Blooming Assessment," in *Proceedings of Australasian Conference on Computing Education, Melbourne, Australia, ACM Press, New York,* 2000, pp. 158-162.

[22] U. Fuller, C. Johnson, T. Ahoniemi, D. Cukierman, I., Hernán-Losada, J. Jackova, E. Lathinen, T. Lewis, D. M. Thompson, C. Riedesel, and E. Thompson, "Developing a Computer Science-specific learning Taxonomy," *ACM SIGCSE Bulletin*, vol. 39, no. 4, pp. 152-170, 2007.

[23] J. B. Biggs, and K. F. Collis, K.F., *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome*. New York, NY: Academic Press, 1982.

[24] Bloom, B.S., Engelhart, M.D., Furst, E.J., Hill, W.H. & Krathwohl, D.R. (1956). *Taxonomy of Educational Objectives: Handbook 1 Cognitive Domain*. Longmans, Green and Co Ltd, London.

[25] L. W. Anderson, D. R. Krathwohl, P. W. Airasian, K. A. Cruikshank, R. E. Mayer, P. R. Pintrich, J. Raths, and M. C.Wittrock, *A taxonomy for learning and teaching and assessing: A revision of Bloom's taxonomy of educational objectives*. New York, NY: Addison Wesley Longman, Inc., 2001.