“I wish I could rank my exam’s challenge level!”: An Algorithm of Bloom’s Taxonomy in Teaching CS1

Mohsen Dorodchi
Department of Computer Science
UNC Charlotte
Charlotte, NC, USA
Mohsen.Dorodchi@uncc.edu

Nasrin Dehbozorgi
Department of Computer Science
UNC Charlotte
Charlotte, NC, USA
ndehbozo@uncc.edu

Tonya K. Frevert
College of Computing & Informatics
UNC Charlotte
Charlotte, NC, USA
tfrevert@uncc.edu

***Abstract*—Designing course activities in harmony with class assignments and tests while providing both adequate challenges and appropriate content progression is critical in introductory programming courses (CS1). Such fine-tuned practices help students build the right mindset to perform better and prevent potential discouragement due to disharmonious test challenges. In this study, we apply levels of the cognitive domain of Bloom's Taxonomy to determine the appropriate challenge level of test questions in CS1. Bloom's Taxonomy has been widely referenced by researchers as a benchmark for assessment of students' learning. Our proposed approach serves two purposes: 1) exposing students to a well-defined set of assessment tests to challenge them based on different levels of Bloom's Taxonomy; and 2) identifying the student's' difficulty areas to redesign and/or reorganize the class activities accordingly. For this purpose, a rubric is developed to classify questions based on cognitive domains of Bloom's Taxonomy. We applied the developed rubric to evaluate three semester tests and design a final exam. In designing the final test, we aim to challenge students' skills in a predetermined proportion and combination that maps onto the levels of Bloom's Taxonomy. After each test, students’ problem areas were identified and related class activities were adjusted to address these weaknesses. Preliminary analysis of student grades shows the effectiveness of this method.**

Keywords*—*Bloom’s Taxonomy (BT), cognitive domain, challenge level, assessment, introductory programming, CS1

I. INTRODUCTION

Computer science (CS) educators report that students’ level of satisfaction in learning increases when they feel they have been challenged [1]. Challenges taken to an extreme, however, can discourage students if the challenge they are facing blunts their progress [1]. Therefore, it is essential to provide students with the appropriate level of challenge that stretches their abilities and understanding of the course concepts, yet does not make them feel frustrated. Students’ practice and assessment at the appropriate challenge level is particularly critical in courses such as CS1, which have historically faced retention issues.

To determine the “proper” challenge level of a test or an activity, certain factors need to be considered. First, it is important to note that an activity’s “challenge level” is a relative concept that varies by context and/or situation, such as repetition of materials or similar examples. Second, there should be some metrics in determining the level of challenge related to different contexts. For example, in CS1, how should we determine the level of challenge of a test? One of the most cited tools consistently shown to improve the quality of assessment is the cognitive domain of Bloom's Taxonomy (hence referred to as “BT” in this paper) [11]. In this study, we use BT to determine the appropriate “challenge level” of test questions and propose a method to evaluate and track students’ activities based on their test performance. For this purpose, we derived an assessment instrument—a rubric—from the literature [9, 10, 11] examining the cognitive domain of BT in order to assess the level of difficulty of activities and problems in CS1. In the following sections, we discuss levels of the cognitive domain of BT, determine appropriate BT-based meta-levels, and create a rubric to apply these levels.

II. THE COGNITIVE DOMAIN OF BLOOM’S TAXONOMY

BT has been adopted by numerous educators across disciplines as a valid benchmark to assess the level of learning in a specific subject [2]. The hierarchical structure of the cognitive domain in BT represents the student’s depth of learning in a given subject [2].

The original version of BT was published in 1956, under the title “Taxonomy of Educational Objectives: The Classification of Educational Goals, Handbook 1, Cognitive Domain” [13]. This version of BT begins with the “knowledge” level, which indicates the student’s lowest level of learning (i.e., remembering facts [2], observation and recall of information learned [6]). The next level is “Comprehension”, the ability to understand the information and the ability to translate that information into a different context [6]. “Application” comes next, which is the ability to apply the information in a concrete situation or new context by using the skills and knowledge students have learned [6, 8]. Following the “application level”, students are required to be able to break down a problem into smaller parts and identify relationships (“Analysis”) [2], combine the different parts together to make a new entity by relating knowledge from several areas and use of old ideas to create a new one (“Synthesis”) [6], and evaluate possible solutions or ideas for a given problem and make judgments (“Evaluation”) [2, 6]. Forty-five years later, a revised version of BT was published by Anderson and Krathwohl, in 2001, which is referred to as the revised Bloom’s Taxonomy (hence referred to as “ BT^R” in this paper) [13]. They believed that the original taxonomy (BT) serves more as a measurement tool while the revised taxonomy (BT^R) addresses more dynamic concepts of classification by using verbs and nouns to label categories and

978-1-5090-5920-1/17/$31.00 ©2017 IEEE

sub-categories. Another difference is that in the revised taxonomy (BT^R) the authors believe that "Create" is the most complex level which should come at the end, while in the original taxonomy (BT), "Evaluation" is the last level [13]. In both versions, however, the levels progress from simple to complex and the goal is to scaffold students from the lowest level of BT to the highest level possible [2]. In the following section, we provide an overview of our method in the application of the cognitive domain to determine proper challenge levels of CS1 activities.

A. Meta-levels of the cognitive domain

Although BT is widely used in assessing students' learning, drawbacks exist in applying it to the field of computer science [3]. Consistent interpretation and application is especially difficult in introductory programming courses [12]. To overcome these weaknesses, some researchers have made modifications to BT to adapt it to CS. In one study, Fuller et al. suggest a two-dimensional matrix adopting Anderson and Krathwohl's version of BT^R [3]. They propose a two-dimensional view of BT^R, which categorizes the levels into two general groups of "producing" and "interpreting". Therefore, it presents the "Remember", "Understand", "Analyze", and "Evaluate" levels as "interpreting" on the horizontal axis while the "Apply" and "Create" levels are included in "producing" on the vertical axis. They argue that a new horizontal row named "none" needs to be inserted under "apply" and "create" since some learning may not produce anything [3]. In this way, this two-dimensional matrix visualizes the semi-dependency of different levels of BT^R with each other.

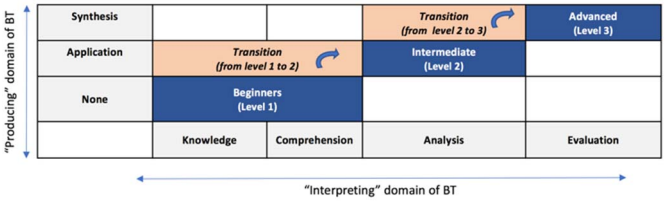
In another study [4], Christopher et al. propose a meta-level structure of BT which has three higher levels: 1) "beginner", in which the learner simply memorizes and understands the concept at a basic level; 2) "intermediate", in which the learner uses and applies what has been learnt; and 3) "expert", in which the learner is able to create, design, and critique something. Moreover, they divide each of these meta-levels into two phases—"produce" and "explain"—to determine each of the original BT levels within these new attributes [4]. These categorizations are elaborated in Table 1.

TABLE 1. META-LEVEL STRUCTURE OF BT [4]

Meta-levels	Phases	BT levels
Beginner	Produce	Knowledge
	Explain	Comprehension
Intermediate	Produce	Application
	Explain	Analysis
Expert	Produce	Synthesis
	Explain	Evaluation

Although the meta-level model has the advantage of defining the mastery level over the cognitive domain of the original taxonomy, it still lacks a consistent pathway through levels of BT for CS1. Such realistic progression is required in exploration of necessary levels of BT to ensure the proper challenge level for students. Moreover, we support the argument of [3] in which, the "remember" level is considered under the "interpret" category rather than "produce" phase (as mentioned in [4]). To overcome the mentioned issues of previous studies, we combined the two views of the taxonomies mentioned in [4] and [3] into a 2-dimensional taxonomy. We also took a further step and added some transition phases between the three meta-levels as shown in Fig. 1. The two dimensions of the matrix in Fig. 1 indicate two scopes of competencies students should achieve throughout a given CS course: 1) being able to understand and interpret and 2) being able to produce and create programs. In this matrix, different levels of BT are distributed to reflect the progression sequence that is critical to follow in CS1.

Fig. 1. Progression through levels of the cognitive domain of BT



We considered the original taxonomy as a basis of the proposed matrix since we believe that in an introductory programming course, "Evaluation" is the most complex level, which comes after "Synthesis". We argue that students in introductory programming courses are not expected to perform in the "advanced" level of BT because they do not obtain the required knowledge or expertise to critique and test the performance of programs or codes based on standard methodologies ("Evaluate").

Therefore, the proposed BT matrix advises us as to the proper order of course content (or tests levels) to be covered to achieve fundamental fidelity across levels and during transitions between levels in an introductory programming course. For example, a CS1 student is considered to have transitioned from beginner level to intermediate level when she is able to apply what she comprehended to the analysis of a program.

III. OUR METHOD

In our study, we pursue two overarching goals: 1) analyzing test questions to uncover students' weaknesses and improve their learning by focusing class activities on their weaknesses and 2) analyzing students' performance on different tests to design an appropriate final exam. In our CS1 course, we administered three semester tests and a final exam (along with periodic lab tests). Each of our semester tests included about 40 multiple choice questions (MCQ). The distribution of topic coverage in each test is shown in Table 2.

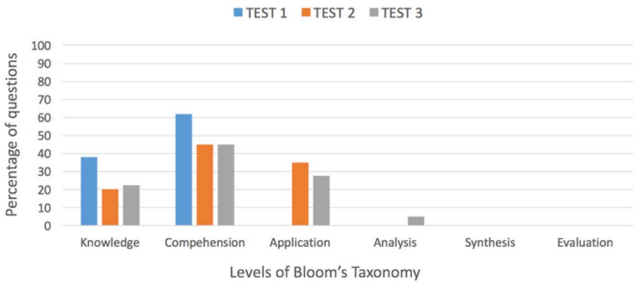
TABLE 2. COVERAGE LEVEL OF CS1 COURSE TOPICS WITH TESTS (THE * INDICATES LESS EMPHASIS AND THE ** INDICATES MORE EMPHASIS)

	Test 1	Test 2	Test 3
Variables, assignments, operators	**	*	*
Decision making	*	**	*
Loops		**	**
Methods		*	**
Arrays			**

After administering each test, we analyzed the test questions and students' answers to determine the appropriate class activities before the next test. For this purpose, we did the

following steps: 1) We created a rubric by extracting the features of each level of the BT, such as the required skills, keywords, and concrete examples from the literature [5, 2, 7, 12] as illustrated in Table 3. 2) We applied this rubric to categorize the test questions to the appropriate cognitive domain level. In the process of categorizing the questions, we found a fine distinction between the "Comprehension" and "Application" levels. During "Comprehension", the students have previously seen the solution of a similar problem, yet during "Application", the process/algorithm of solving the problem is familiar for students though they have not seen the solution before [12]. 3) The categorized questions were verified by four instructors and teaching assistants to ensure the coherency of the mappings with the levels of the cognitive domain. The students' previous exposure to class activities and pre-tests were considered to properly differentiate between the "Comprehension" and "Application" levels. As a result, by applying the rubric we were able to determine the coverage level of each test based on the cognitive levels of BT as illustrated in Fig. 2.

Fig. 2. Mapping the tests to the levels of Bloom's Taxonomy



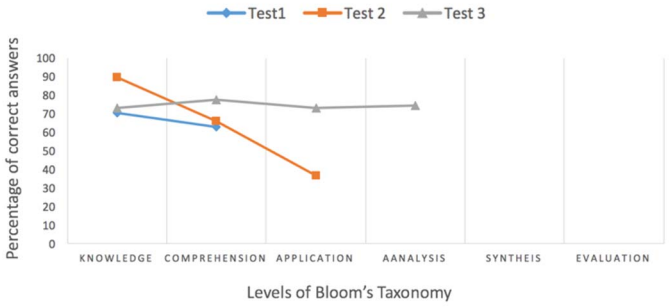
4) Based on the students' answers to each test question, we could identify the cognitive levels in which the students had difficulties. Therefore, we were able to design some class activities to address the uncovered weaknesses. 5) Finally, considering the students' performance on the three tests over the semester, we designed the final exam with appropriate levels of challenge for students while maintaining smooth transitions between the cognitive levels. Therefore, the final lecture and lab exams include the necessary variation in both depth and breadth of content, striking a balance between students' level of mastery to cover the necessary course materials and properly assessing students' knowledge. In the following section, analysis of the students' performance based on our proposed method is provided.

IV. Results

To analyze our method, we applied it to three consecutive tests (before the final exam) in Spring 2017 and analyzed the students' performance on these three tests. Using the progression matrix of Fig. 1, we can confirm that Test 1 is a beginner level test since it mainly includes the "remember" and "understand" domains. For Test 2, however, we observe a sudden increase in the "apply" level, which categorizes this test under the "transition to intermediate" level. Lastly, for Test 3, we observe a smoother distribution of cognitive levels, including a few "analyze" problems, which thus placed this test under the category of "intermediate" level.

After each test, we tallied the number of students who had the correct answer to each test question. Accordingly, we were able to identify deficits in students' learning and/or determine possible improvements through cognitive domain of BT. The students average scores on these three tests were: 66% (Test 1), 61% (Test 2), and 76% (Test 3). Fig. 3 shows how far the students progressed through the cognitive domain from one test to the next test.

Fig. 3. Student progression through cognitive domain



It can be inferred that as students start "applying" more in-depth concepts, they tend to forget ("not remember") some of the basic theoretical concepts they have already learned (as Fig. 3 shows decrease in the "knowledge" average score in Test 3 compared to Test 2). This decrease confirms that we need an iterative progression through cognitive domains of BT because once students "pass" a level, it does not guarantee they maintain it for a long time. Therefore, iterative practices emphasizing all concepts are required for students, especially in CS1. Further, we uncovered challenging questions in Test 2 ("application" questions) and infused more relevant class activities that resulted in improved overall students' performance on Test 3. After students took Test 3, we noticed significant improvement in their performance after implementing this method properly between Test 2 and Test 3. The hypothesis test for paired observation was conducted with result of very low p-value and Cohen's effect size value of $d = 0.765$ (for sample size of $n = 84$) which suggested a moderate to high practical significance. Results of Test 3 confirmed that students are in transition level between intermediate and advanced. At this point, students are ready to practice more on "application" of their knowledge in "analysis" and "synthesis" of larger programs.

In conclusion, designing proper challenge levels of tests helps students with smooth transition into deeper learning. We consider this finding as a very significant accomplishment which results in overall enhancement of students' performance and learning. Our primary analysis of Spring 2017 students' grades shows improvement compared to previous semesters as well. Toward the final exam, we introduced more appropriate adjustments to questions based on their challenge level and fine-tuned the class activities with the same level of challenge.

V. Conclusion

In this study, we applied Bloom's Taxonomy to determine the appropriate challenge level of test questions in CS1. Our proposed approach helps instructors identify students' difficulty

areas in a timely manner to redesign and/or reorganize subsequent class activities accordingly. For this purpose, a rubric is developed to classify questions based on levels of the cognitive domain. We applied this rubric to analyze the tests during the semester and to design the final exam. After each test, we identified the problem areas of students across the different levels and related activities were thus adjusted to compensate for the identified weaknesses. Preliminary analysis of student grades showed that while students had minor improvement after Test 1, they demonstrated major improvement after Test 2. We believe that applying our algorithm ,which was presented as a rubric to apply BT, helps educators produce higher quality tests that challenge students at the appropriate level to make learning CS1 concepts an enjoyable experience. This pedagogical approach provides an opportunity for CS instructors to determine their students' pace of learning and skill development throughout the semester and use that information to provide appropriate activities in a timely, responsive manner to improve student performance.

TABLE 3. RUBRIC INCLUDING KEYWORDS, REQUIRED SKILLS, AND SAMPLE QUESTION FOR DIFFERENT LEVELS OF BLOOM'S TAXONOMY [5, 2, 7, 12]

BT levels	Associated Actions	Description of Skills at BT Levels	Sample Questions
Knowledge	List Recall Recognizing {identifying} Recalling {retrieving}	-Student can list related commands/concepts. -Identifying a particular construct in a piece of code	-Questions mainly about the theories, syntax, and function of frameworks of structured programming (mainly the material that were explicitly covered in teaching material) -listing the arithmetic operations in order of precedence
Comprehension	Interpreting {clarifying, paraphrasing, representing, translating} Exemplifying {illustrating, instantiating} Classifying {categorizing, subsuming} Summarizing {abstracting, generalizing} Inferring {concluding, extrapolating, interpolating, predicting} Comparing {contrasting, mapping, matching} Explaining {constructing models}	-Student can explain what the command/concept means. -Student can apply an example to a similar problem.	-Giving a piece of code asking the behavior of code or algorithm. -Finding out the output of the code. -Translating an algorithm from to another form -providing an example of algorithm or a concept -given a piece of code, students are asked to identify the constructor(s) by writing the constructor signature
Application	Respond Provide executing {carrying out} implementing {using}	-Student can list cases when the command/concept can be used. -Student can apply an example to a different problem.	-Questions which expect students to apply prior knowledge to new problems. Three types of questions can be used: -Giving a piece of code including (process, decision making and repetition) asking the students to find an equivalent command for the given code. -A logical diagram is given and asking to write the commands to execute the diagram (or flowchart). -Gap filling in a given piece of code. (These types of questions improve students' sub-problem solving and are considered to be more challenging).
Analysis	Differentiating {discriminating, distinguishing, focusing, selecting} Organizing {finding coherence, integrating, outlining, parsing, structuring} Attributing {deconstructing}	-Student can explain the meaning of the command/ concept in its context, why it is there. -breaking a program down into its components -ordering the component parts to achieve the program goal -identify important or unimportant components of a given program	-Finding logical error and bugs in the given code. Breaking down the code into logical parts and analyzing them. -Filling a missing part of a given code (this was in apply level too). -How a given method differs from other methods in a given class
Synthesis	Assemble Create Generating {hypothesizing} Planning {designing} Producing {constructing}	-Student can use the command/concept in problem solving without an example.	-Questions which expect the students to combine pieces of code or code from scratch to solve a given problem. -Usually the problems at this level are new for students so that they avoid pattern matching.
Evaluation	Determine Reflect Checking {coordinating, detecting, monitoring, testing} Critiquing {judging}	-Student can ensure the correct use of the command/concept. -ensuring that the program satisfies the requirements using testing strategies -critiquing a piece of code based on coding standards	-This level considered to be advanced and not applied in our method for CS1.

REFERENCES

[1] H. Walker. 2011. How to Challenge Students, Classroom Vignettes, ACM Inroads, Vol. 2, No. 3.

[2] Bell, John T., Folger S.1995. The Investigation and Application of Virtual Reality as an Educational Tool, Proceedings of the American Society for Engineering Education, Annual Conference.

[3] Fuller, U., Ursula, J., Colin, G., Ahoniemi, et. al. 2007. Developing a Computer Science-specific Learning Taxonomy, ACM SIGCSE Bulletin, 39 (4), 152-170.

[4] Manaris, C. W., Stalvey B., Roxann H.2008. Bloom's Taxonomy Revisited: Specifying Assessable Learning Objectives in Computer Science, SIGCSE '08 Proceedings of the 39th SIGCSE technical symposium on Computer science education, 261-256.

[5] Chatzopoulou, D. I., & Economides, A. A. (2010). Adaptive assessment of student's knowledge in programming courses. Journal of Computer Assisted Learning, 26(4), 258–269. http://doi.org/10.1111/j.1365-2729.2010.00363.x

[6] Hamilton, M., & Souza, Daryl D, S. S. (2007). A Taxonomic Study of Novice Programming Summative Assessment, (Ace), 147–156.

[7] Heer, R. (2012). A Model of Learning Objectives. Iowa State University, 1–3. http://doi.org/10.1207/s15430421tip4104_2

[8] Oliver, D., Dobele, T., Greber, M., & Roberts, T. (2004). This Course has a Bloom Rating of 3.9. Proceedings of the Sixth Australasian Conference on Computing Education, 227–231.

[9] Alaoutinen, S., & Smolander, K. 2010. Student self-assessment in a programming course using bloom's revised taxonomy. Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education - ITiCSE '10, 155. http://doi.org/10.1145/1822090.1822135

[10] Chatzopoulou, D. I., & Economides, A. A. 2010. Adaptive assessment of student's knowledge in programming courses. Journal of Computer Assisted Learning, 26(4), 258–269. http://doi.org/10.1111/j.1365-2729.2010.00363.x

[11] Hamilton, M., & Souza, Daryl D, S. S. 2007. A Taxonomic Study of Novice Programming Summative Assessment, (Ace), 147–156.

[12] Thompson, E., Luxton-Reilly, A., Whalley, J. L., Hu, M., & Robbins, P. 2008. Bloom's taxonomy for CS assessment. Conferences in Research and Practice in Information Technology Series, 78(January), 155–161.

[13] Krathwohl, D. R., Anderson, L. W., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., Wittrock, M. C. 2002. A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives, (Abridged Edition). Theory Into Practice, Volume 41, Number 4. http://doi.org/10.1207/s15430421tip4104_2