US-China Education Review B 1 (2012) 41-49
Earlier title: US-China Education Review, ISSN 1548-6613

A CAD (Classroom Assessment Design) of a Computer Programming Course

Nazir S. Hawi
Notre Dame University-Louaize, Zouk Mosbeh, Lebanon

This paper presents a CAD (classroom assessment design) of an entry-level undergraduate computer programming course "Computer Programming I". CAD has been the product of a long experience in teaching computer programming courses including teaching "Computer Programming I" 22 times. Each semester, CAD is evaluated and modified for the subsequent semester. The assessment targets have been determined through the course's objectives and sub-objectives. Consequently, grades that resulted from CAD described how much of the course content students had acquired. Also, technology impacted CAD in several ways, such as using the development tool itself, and/or using an e-learning system. CAD in its current state includes three major categories: appreciation, assignments and exams. All exams have been developed within Bloom's six-level cognitive hierarchy. This paper includes examples of all assessment tools mentioned above. The main lesson learned is that the more effort is invested in improving CAD, the more useful it becomes.

Keywords: CAD (classroom assessment design), computer programming, higher education, learning outcomes

Introduction

This paper presents a CAD (classroom assessment design) of an entry-level undergraduate computer programming course "Computer Programming 1" at Notre Dame University, Louaize. This design has been the product of a long experience in teaching computer programming courses. This experience includes teaching "Computer Programming 1" 22 times. The first time the author taught the course was in fall, 2001 and the last time was in spring, 2011.

Semester after semester, the course's assessment has been tried. As a result, it has been modified several times. Assessment was based on the author's perception of which learning targets should be assessed and how to assess them. The assessment targets were determined through the course's objectives and sub-objectives. Each assessment generated a lot of data that were used to evaluate it. Evaluation was used to improve assessment. With each implementation, assessment produced grades that were believed to be better informative measures that describe how many of the course contents students have acquired. It is worth mentioning that more efforts could have been placed to study the information generated by assessment which had the instructor more time. The more efforts are invested in improving the assessment design at hand, the more credible are the reported measures of students' achievement.

Course Objectives

Classroom assessment stems from the course objectives. The objectives as stated in the "Computer

Nazir S. Hawi, Ed.D., assistant professor, Computer Science Department, Notre Dame University-Louaize.

Programming 1" course syllabus are as follows:

(1) To learn the fundamentals of computer programming;

(2) To develop programming skills to build small applications using fundamentals of computer programming;

(3) To use the most recent tools and features of software development;

(4) To prepare students to transfer their knowledge of the fundamentals of computer programming from one programming language to another;

(5) To prepare students to cope with the constantly changing field of computer programming;

(6) To assess students with and without using a software development tool.

There are six objectives in all. This number is in resonance with a common conviction that there should not be too many course objectives (Popham, 1995). In addition, they are broad, and most of them are measurable. Learning programming encompasses all levels of abstraction plus psychomotor skills. Based on the course objectives, only cognitive assessment is required to measure mental skills. Psychomotor assessment that measures physical skills is not required. An example of physical skills is the proper use of keyboarding for keying in and editing code. Computer programming encompasses deep understanding and reasoning cognitive processes, such as to analyze, to evaluate and to create. These processes should be a part of classroom assessment.

CAD (Classroom Assessment Design)

The assessment design in its current state includes appreciation, assignments and exams. Appreciation is subdivided into:

(1) In-class participation, such as asking questions, answering questions, giving examples, offering clarifications and engaging in discussions;

(2) Demonstrating ability to develop or participate in the development of an application in-class, on paper or using the class computer;

(3) Unannounced quizzes that: (a) assess just in time comprehension with the aim of triggering interest in topics under focus and promoting paying attention in class; (b) encourage students to study and practice outside the classroom on a daily basis; and (c) verify whether the students who submitted their assignment are the ones who made them;

(4) Dynamic quiz-taking that is used to give students a second chance to study the material which a significant number of students failed on an unannounced quiz. The second attempt includes different questions more often than not;

(5) Discussion board of the e-learning system blackboard that involves students in posting comments on a given thread or in discussing others' comments using references, such as textbooks or reliable Internet sources;

(6) Unassigned creative work produced outside the classroom and presented in class or during the instructor's office hours;

(7) In-class use of textbook and note-taking.

All appreciation components are used for ongoing multidirectional feedback. This multi-directional feedback informs the students, the instructor and the administration. Appreciation activities are an integral part of the teaching and learning process (McMillan, 2004). They reassure students who understand the learning targets and they alert those who do not.

Formal assessment includes assignments and exams. There are three assignments. The first assignment is

carried out based on individual basis. Each student is asked to develop applications on individual basis. This assignment falls in the application category of the cognitive domain. The educational goal is to engage each student in an activity to learn to think independently. For this end, a student uses programming concepts learned in class, but in a new situation. Each student is required to find a software solution to three problems of varying degree of difficulty: easy, intermediate and difficult. To submit this assignment to the instructor, the three solutions are saved in one folder. The latter is compressed and uploaded via the assignment tool of the e-learning system blackboard. The second assignment is carried out based on group work which is based on the development of applications. This type of assessment is repeated in the next two courses in the sequence of computer programming courses. The educational goal is to engage students in an activity where they work collaboratively with others. Collective work helps students to adapt to other classmates and their ways of communication, to be exposed to others work practice and learn about others strengths and weaknesses. This assignment and similar subsequent ones prepare students in all respects for the senior project which follows the sequence of computer programming courses. Students willing to work together should submit their names within a specified time limit. Each group chooses three problems from a set of problems containing three levels of difficulty: easy, intermediate and advanced. From each difficulty level, one problem is selected. When a group completes an assignment, it submits the work via blackboard's assignment tool. A follow-up quiz is administered to make sure that all members of a group had participated in the development of every problem. This assignment falls in the application category of the cognitive domain. Students use programming concepts in new situations. The third assignment is an individual work based on an oral in-class presentation of a program, or of a computer programming concept. The purpose of this assignment is to enable computer programming students to show the knowledge of a topic they acquired by using the proper terminology and specific details and learn the skills needed to present the chosen topic including time management. The teacher presents to the students a list of topics to choose from. Students pick from this list topics of interest to them. Then, the teacher is consulted. If the instructor has no objection, the topic is reserved on the basis of first come first served. Else, the student is advised to name another choice. When a student is ready to present a programming concept, the teacher and student set up an oral in-class examination session of 10 minutes. If the number of students is large, then the presentations are delivered in slots during the instructor's office hours. This assignment falls in the comprehension category of the cognitive domain. Students should show that they understand the programming concept under focus or can interpret programming instructions. In addition, such activity is expected to bring about conversations on the topic.

As for exams, the assessment of the computer programming course at hand encompasses two tests and a final exam. All exams have been developed based on the revision of Bloom's six-level cognitive hierarchy (Anderson, 2001). Each exam includes a combination of different types of questions. Possible question types are: binary choice, multiple choice, completion, defining terms, writing code, analyzing code, correcting errors, finding output, explaining given code, explaining programming concepts, reordering instructions to make a code block workable and finding and discussing possible scenarios.

Like appreciation, exams and quizzes have been developed and employed to inform the students, the instructor and the administration. They show what and how much students learned of the course's objectives. Also, they show how much the instructor's instruction was effective. Areas of weakness should be remedied. This is essential for everyone. In the short term, students should fill in gaps to improve their knowledge and demonstrate it in the comprehensive final exam. In the long term, students need to be prepared for related

subsequent courses and for their future career. Further, exams and quizzes are essential for the instructor, especially for the purpose of teaching the course again. Of course, exams generate grades that are all what a third party needs, such as a registrar to fill transcripts and parents to obtain an idea of their children's progress.

Exams are constructed to assess knowledge, understanding and reasoning. All the examples that follow are drawn from programming concepts, variables and methods, taught first in the course. The reason for this is to keep the examples as simple as possible for educators who are unfamiliar with the teaching and learning of computer programming. Knowledge of computer programming syntax, keywords and terminology concepts is assessed with selected-response items, completion items and short-answer. Selected response items include binary choice items, such as number 1 below and multiple choice items, such as number 2 (see Appendix). Completion items are incomplete statements used to measure how well students recall computer programming keywords, such as item 3a (see Appendix), terminology and concepts, such as item 3b (see Appendix). Keyword items may include items about Handles, As, Dim, Public and Sub, just to name a few keywords. Terminology items may include items about OOP (object-oriented programming), debugging and IDE (integrated development environment). Concept items may include items about memory, classes and methods.

Comprehension of concepts or procedures is assessed by asking questions that require students to respond with short-answer questions, such as numbers 4 and 5, completion items, such as number 6, binary choice, such as number 7, error detection, such as number 8, explanation, such as numbers 9 and 10 and comparing concepts, such as number 11 (see Appendix). Number 8 assesses the ability of students to find errors (see Appendix). Finding errors is a thinking skill that requires analysis. There are three types of errors: syntax, execution and logic (Willis & Newsome, 2008). Number 11 includes short-answer questions that require students to supply brief responses. This question assesses the ability of students to compare two computer programming concepts. Comparing is a thinking skill that measures understanding of the computer programming concepts functions and methods in the first question and integer and single data types in the second question.

Application is assessed by asking questions that require students to use what they learned about a programming concept in a new situation (McMillan, 2004), such as numbers 12 and 13. Number 12 assesses the student's ability to construct GUI's (graphical user interfaces) under new requirements. Similarly, number 13 assesses the student's ability to create an event and a method based on existing knowledge of these concepts in new situations.

The second set of questions assesses the student's ability to analyze, evaluate and create computer programming concepts. The latter three cognitive processes assess the deep understanding and reasoning which forms the other part of the revision of Bloom's taxonomy (McMillan, 2004). Number 14 assesses the student's ability to analyze code. The first item of number 14 is required to distinguish the different parts of an event handler's header and identify the role of part. The second item of number 14 requires the student to find the correct order of the given instructions to produce a coherent and workable code block. Number 15 assesses the ability of students to check a code block. As a direct result of the checking, the student should be able to critique the novice programmer claim. Number 16 assesses the student's ability to evaluate code. The student is required to check the code block and determine the output as a consequence. Numbers 15 and 16 assess the student's ability to evaluate code, a cognitive process that is a part of the deep understanding and a reasoning part of the revised Bloom's taxonomy (McMillan, 2004). Number 17 assesses the student's ability to create code. Each item of number 17 requires the student to produce a code block for a specific purpose (see Appendix).

CAD Improvement

Although the assessment of the computer programming course at hand shared common characteristics for some semesters, subsequent assessment was improved based on data derived from a current assessment. Over the years, the author developed the habit of making necessary amendments to his courses' assessments batteries when he starts gathering data, even if there was little hope to teach the course again. Changes include: question's wording, weight and type; redistribution of points over parts of a question; redistribution of points over an exam's questions; exam duration; number of questions; inclusion of new question types; and removal of tested question type that proved not useful. Also, changes affected the number of quizzes, assignments and exams, and their types, administration and management.

The major reason for the aforementioned changes is to construct the fairest assessment design possible. An example of test bias is when a test features questions just about writing code. Such a test favors learners who can retain instructions in memory. It promotes for rote learning, by neglecting the other thinking functions. Also, it works against the major improvements made to code editors, such as the IntelliSence feature. The latter lists a class's members while writing code in the editor (Willis & Newsome 2008). Further, it does not reflect the learning strategies followed in the classroom and outside it. The same applies on tests that favor just analysis.

Conclusions

This paper presented a CAD of an entry-level undergraduate computer programming course "Computer Programming 1" at Notre Dame University, Louaize. The course's life time is 15 weeks. The assessment design as presented is the result of several years of experience. In its current state, it includes appreciation, assignments and exams. While appreciation and assignments were illustrated with general guidelines, exams were illustrated using concrete examples. Although the author finds this assessment design satisfactory, he believes that there is plenty of room for improvement.

Recommendations

Assessment is a very important aspect of teaching and learning. However, rarely educational institutions include in the process of hiring teachers the criterion of being skilled in learning assessment. As a result, instructors administer tests to label their students performances with the most concise form, grades. In such cases, courses follow a one way path where teaching is the input and grades are the output. This is an over simplified way of looking at the issue. Probably, people who work in software development are amongst those who appreciate most the importance of testing. Testing is an essential phase for them. There are specialized people who go through cycles of different types of testing until all requirements are met and all disparate bugs are fixed. In education, employing learning assessment with the intention to fill in learners' deficiencies and improve one's teaching is not widespread especially in higher education. Despite of this gloomy view, there are many educators who are fighting for better assessment means and this conference is evidence.

Office hours should be used for remedial instructional activities. Teachers spend their office hours expecting students with deficiencies to show up and ask for help. Teachers wait for their students to show up live their office hour time like Vladimir and Estragon did by waiting for Godot in the play "En attendant Godot" by Samuel Beckett. In higher education, many students with deficiencies do not ask for help. At school, parents advise their children to ask their teachers for help. Who can play this role in higher education? Academic advisor should. Students do not ask for help for several reasons. They might not know where to start

their enquiries content wise, because it might unveil more ignorance to the teacher than the assessment has already done. Also, students might be afraid of their teacher's reaction based on previous negative experiences with the same or other teachers. Teachers can go the extra mile and have students with deficiencies get the appropriate help until they meet the course's instructional targets. The implementation of such after assessment plan pushes teachers and students to leave their comfort zones. However, there will be a quality shift from individual comfort zones to an ethical zone approved by the community.

References

Anderson, L. W. (2001). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. New York: Longman

McMillan, J. H. (2004). *Classroom assessment: Principles and practice for effective instruction* (3rd ed.). Boston: Pearson.

Popham, W. J. (1995). *Classroom assessment: What teachers need to know*. Boston: Allyn and Bacon.

Willis, T., & Newsome, B. (2008). *Beginning Microsoft Visual Basic 2008*. Indianapolis, I. N.: Wiley Publishing.

Appendix

1. Binary choice

   a. T F The shorthand version of $n = n + 1$ is $n + = 1$

   b. T F The data type that holds values that represent dates and times is time

2. Multiple choice

   a. Which of the following is not a data type:

   (A) Byte (B) Single (C) Double (D) Triple

   b. Which of the following is a Boolean value:

   (A) "True" (B) 'True' (C) True (D) T

3. Completion items

   a. A statement that declares and allocates storage for a variable is _____.

   b. A statement that is used to declare explanatory remarks in the source code of a program _____.

4. Defining terms

   a. Define concatenation

   b. Define method

5. Short-answer questions

   a. Methods are essential for a program for two reasons which are:

   _____

   _____

   b. Boolean are important for programs when:

   _____

6. Completion items

   a. A self-contained block of code that "does something" is called _____

   b. Something that you store a value in is called _____

7. Binary choice

   a. T F Numbers, such as 34 and 6,789 are best stored in an integer variable.

b. T F When a decimal is stored in an integer variable its decimal part is removed and its integer part is stored as it is

8. Error detection

a. Identify the syntax errors in the following code line, then correct them.

Private function calculate (ByVal num Az Doubel) Az Singel

b. Find the errors in the following instruction that calls the function in part (a).

Calculate (“2”)

9. Explaining given code blocks

a. Private Sub btnConcatination_Click (sender, e) Handles btnConcatination. Click

Dim strCourseName, strCourseNumber, strBoth As String

strCourseName = “Computer Programming 1”

strCourseNumber = “CSC 216”

strBoth = strCourseName & “,” & strCourseNumber

MessageBox.Show (strCourseNumber, “What happened?”)

End Sub

b. Dim strInput as string

strInput = txtInput.Text

MessageBox.Show (strInput.Length & “Date.Now”)

10. Explaining programming concepts

a. List and explain two reasons for using methods?

b. List and explain three characteristics of variables?

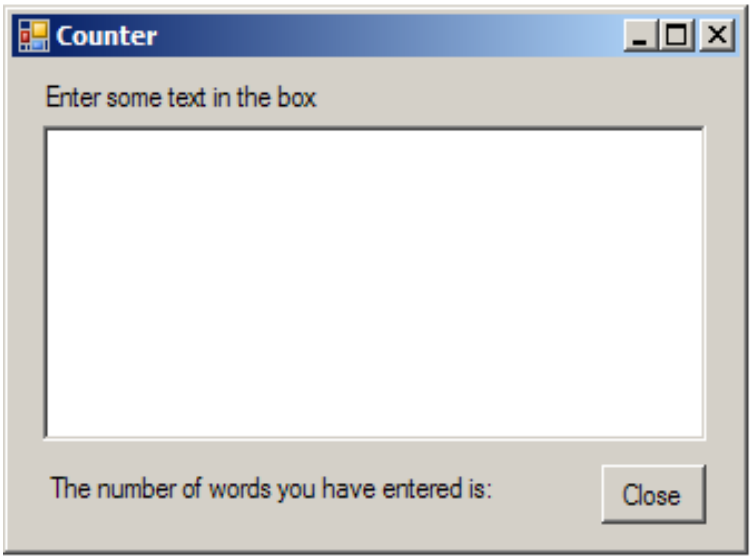
11. Comparing concepts

a. How are function and method different or similar?

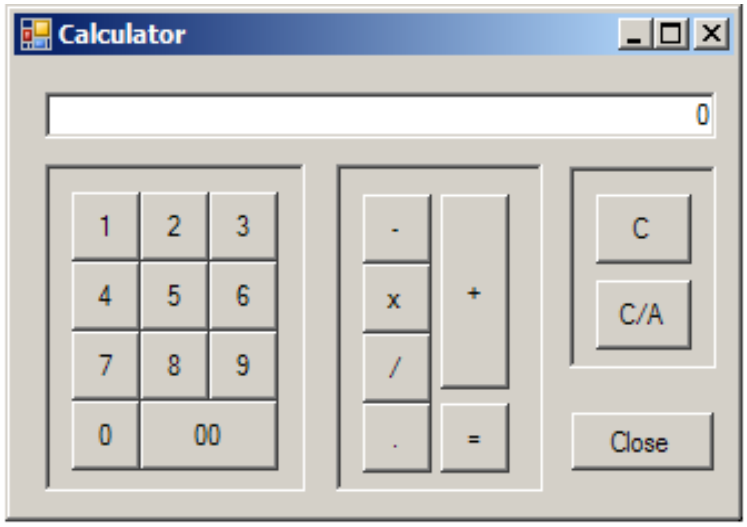
b. How does the Integer data type differ from the Single data type?

12. Create a GUI:

a. For a word counter as shown below



b. For a calculator as shown below



13. Writing code to construct one structure

Create an event that displays the first ten products of a number entered in a text box control. The products are displayed when the user clicks a button.

Create a Say Hello method that greets the user. This method receives the name as an argument.

14. Analyzing code

a. Break the event handler header below into its constituent parts and determine the role of each part with respect to the whole.

```
Private Sub Button1_Click (ByVal sender As Object, _
ByVal e As EventArgs) Handles Button1.Click
```

b. Reorder the following instructions to make the code block workable.

```
End Sub
x = x + 1
IncrementByOne ( )
Dim x as integer
Sub IncrementByOne ( )
```

15. Checking

a. A novice programmer claims that the code block below generates a run time error. Is the novice programmer right and why?

```
Function Find square (ByVal sngNum As Single) As Single
Return sngNum *sngNum
End function
Find square (TextBox1.Text)
```

b. A novice programmer claims that the code block below generates a run time error. Is the novice programmer right and why?

```
Function divide (ByVal sngNumerator As Single) As Single
```

```
Dim sngDenominator
Return sngNumerator * sngDenominator
End function
Divide (120)
```

16. Finding output
    a. Find the output of the following code block:
        1. $x = 0$
        2. $x = x + 1$
        3. $x = 3 * x + 2$
        4. $x = 2 * (x + 3)$
    b. Determine the value that is stored in intNum

```
Dim dblNum As Double = 2.75
Dim intNum As Integer
intNum = dblNum
```

17. Building code blocks that contain several structures
    a. Write a well-organized class that uses two events to illustrate the difference between a local and a global variable;
    b. Write a well-organized class that uses two events to illustrate how an event calls a method.