Teaching and Assessing Quality and Continuous Improvement

Thomas Mertz

Engineering Technology Department
Computer Systems Technology
Kansas State University – Salina

Abstract

This paper describes an attempt to teach and assess students' commitment to quality, timeliness and continuous improvement in a computer software course. The paper discusses continuous improvement and describes the course, the projects assigned to the students and the grading model used to assess them. A summary is given of the students' performance and their perspective of the experience.

Introduction

Criteria 3i of the *2012-2013 Criteria for Accrediting Engineering Technology Programs* states that a desired capability of Engineering Technology graduates is "a commitment to quality, timeliness, and continuous improvement." [1]

Continuous improvement is the ongoing effort to improve products, services or processes. There are a number of formal continuous improvement methodologies, including Six Sigma, Lean and Total Quality Management. [2]

It was not my goal to teach such continuous improvement methodologies. Instead, I had three goals: (1) Address quality, timeliness and continuous improvement as a roughly defined work ethic that conscientious professionals practice daily. (2) Compel, as much as possible, the students to practice it. (3) Do this in a course that I regularly teach.

What I Did

A large project spread over a number of weeks seemed the best vehicle on which to practice "continuous improvement." Therefore, I chose to teach the concept in a three-credit project course required of all sophomores in our program. The course introduces some new material relevant to projects and project management, but is overwhelmingly devoted to completing a "live project" for a local business. This consists of a database application using the Microsoft Access database management system. Students are required to program the underlying application logic using Visual Basic for Applications (VBA).

Course topics related to quality, timeliness, and continuous improvement included lessons on the database development lifecycle and software testing. The students and I discussed continuous improvement in class and what it means for a database applications programmer. I told the

students that I expected them to practice continuous improvement on their projects and I illustrated its benefits with personal anecdotes taken from my industrial experience.

I gave the students three projects of increasing difficulty. The first was a database application that that I had done in advance so that I could give the students printed copies of all the tables, forms and algorithms. They simply had to copy my work to implement and test the application in MS Access. The students were required to submit the application twice for evaluation. After the first submission, I returned the students' graded applications and required that they improve them and resubmit them for a second evaluation.

The second project was similar to the first except that I only sketched its requirements roughly with pencil on paper. The students had to work through the design, implementation and testing of the application. This application also had to be submitted twice for evaluation.

The third project was a live project solicited from a local company. The students completed this project in teams of three. Each team had to work through the entire database lifecycle – interacting with the client to determine her requirements, documenting those requirements, designing the database, implementing and testing the application. Each team submitted its application once for evaluation. I met with each team weekly to review their progress and log the extent to which they "improved" the application as suggested by the client and me.

All three applications were evaluated using the same rubric; one that I had distributed and discussed early in the course before assigning the projects. The rubric applies four criteria:

Database Application Evaluation Criteria		
1	Functionality	How well your application fulfills its user requirements.
2	The "Look"	The aesthetics, clarity, accuracy and coherency of your application's user interface.
3	The "Feel"	The appropriateness and ease of use of your application's user interface.
4	Internal Structure	The structure and documentation of the database and the application's internal code.

Criterion 1, Functionality, was graded according to these standards:

Functionality Standards		
Pts.	Rank	Description
4	Exemplary	Works correctly and is highly robust to user error.
3	Acceptable	Works correctly and is highly robust to user error except for isolated situations of unusual user error.
2	Barely Acceptable	Works correctly for typical usage but is not robust (either fails too often, fails ungracefully or fails for predictable user error).
1	Unacceptable	Works incorrectly for typical usage or does not meet requirements.

For the latter three criteria I judged the students' projects using a precompiled list of indicators. For example, here are some of the indicators used to judge criteria 2 and 3:

- Character fonts, styles and sizes promote clarity and are consistent.
- All user observed objects must use consistent capitalization and punctuation.
- Appropriate form control types (e.g. combo box vs. text field) must be used to improve usability and reduce user error.

A student's grade for these criteria directly reflected the extent to which he or she used the indicators. An "acceptable" project used the indicators; an "unacceptable" one didn't. I reserved the highest grade, exemplary, for those projects that showed commercial quality, i.e. what I would've done had I been paid for it. Here's the general rubric for criteria 2, 3 and 4:

The Look, Feel and Internal Structure Standards		
Pts.	Rank	Description
4	Exemplary	Project appears professionally done and well crafted.
3	Acceptable	Consistent application of the criterion's indicators but the overall project appears amateurish or perfunctory.
2	Barely Acceptable	Inconsistent application of the criterion's indicators.
1	Unacceptable	Minimal application of the criterion's indicators.

The Result

The average rankings for the students' applications are shown below. The students' applications were, on the average, less than acceptable when evaluated the first time. The students were able to improve their applications to acceptable levels for the second evaluation.

Average ranking over 4 criteria and 18 students	Project #1	Project #2
First Evaluation	1.57	1.53
Second Evaluation	3.21	2.64

Assuming that students understand the concept of quality – they certainly can articulate what they consider to be a quality video game or a quality college course – these numbers suggest that they cannot relate the concept of quality to what they do for a college course. For example, their first evaluation of Project #1 was less than acceptable (1.57) even though they had been given detailed specifications and evaluation criteria. Nor did the students seem to learn any lessons from Project #1 that they carried to Project #2.

The third project was completed by six teams of three students each; the one and only evaluation yielded these rankings:

Team	1	2	3	4	5	6	Average
Ranking	4.00	3.67	3.50	3.33	2.50	1.00	3.00

These numbers suggest that the students did carry the lessons of quality and continuous improvement over to the live project. In other words, once the project environment itself more resembled the real world, the students' behavior modeled what is expected in the real world.

Furthermore, students performed better on the live project this year than in previous years. This table shows the rankings of the students' final projects in a previous year when quality and continuous improvement was not explicitly addressed in the course.

Team	1	2	3	4	5	6	Average
Ranking	2.43	2.43	2.29	2.00	1.86	1.86	2.14

At the end of the course, I surveyed the eighteen students using a Likert-type scale on which each student ranked his or her perspective regarding the course and quality and continuous improvement. Here are the results of that survey:

Scale: 1 = Not at all, 2 = Somewhat, 3 = A lot	**Average**
Rate the extent to which this course made you **aware** of the importance of quality and continuous improvement.	2.64
Rate the extent to which the course's **grading model** encouraged your dedication to quality and continuous improvement.	2.64

Conclusion

A project course in database application development was used as a vehicle for teaching students about quality and continuous improvement. Students were taught continuous improvement as an informal work ethic and shown how to practice it in developing software. They were encouraged to practice it for two projects by requiring them to improve their work, for which they would receive a higher grade as a reward. On the third and final project, student teams were to employ continuous improvement in the development of a database application for a local business. Each team's work was reviewed in weekly meetings with the client and the instructor. The instructor assessed the continuous improvement by logging the extent and quality of the team's response to suggestions by the client and instructor.

The overall results were (1) the quality of the final project was greatly improved over previous years and (2) the students indicated that they became aware of and were encouraged to engage in quality and continuous improvement.

Bibliography

1. ABET, "Criteria for Accrediting Engineering Technology Programs, 2012 – 2013," (Accessed 2013). Available WWW: http://www.abet.org/DisplayTemplates/DocsHandbook.aspx?id=3144.

2. American Society for Quality, “Continuous Improvement,” (Accessed 2013), Available WWW: http://asq.org/learn-about-quality/continuous-improvement/overview/overview.html

Biographical Information

THOMAS MERTZ
Associate Professor, Computer Systems Technology, Kansas State University – Salina (tmertz@ksu.edu)