Teaching Error Correction to Core IT Students via Video Supplementary Instruction

MAJ John Syers
Department of Electrical Engineering and Computer Science
United States Military Academy, West Point, NY 10996

Abstract

The introduction to programming can be very difficult for students, particularly those who have no IT background. Understanding and correcting syntax errors is an integral part of programming, yet this topic is often given only perfunctory mention in course curriculums. The goal of this study is to determine whether providing supplementary instruction to students is an effective means of teaching error correction. It also explores the effects this instruction has on instruction in the classroom. The study uses a popular online video website as the means of distribution, and also seeks to determine whether this type of communication is preferable, or whether another form of media should be used. Students were surveyed to gain their feedback on the usefulness of this form of instruction. The results of this study will provide insight to faculty on the relative benefits of investing time in preparing materials that will be primarily used outside of the classroom.

1. Introduction

This study was formulated as a result of trends observed in IT105, Introduction to Computing and Information Technology. The United States Military Academy is somewhat unique in that its core curriculum includes two information technology courses. Cadets take the first course during their freshman year and the second junior year. IT105 introduces cadets to a problem solving process, using Java as the vehicle. Because the goal is for cadets to assimilate the process and programming concepts rather than become programmers, we use the RAPTOR design tool, which greatly reduces the number of novice programmer mistakes[1].

Cadets generally did well on programming assignments gradewise. The assignments were believed to be valid assessments, and so it was reasonable to conclude that the cadets had an understanding of the general programming concepts, but something was lacking. A closer observation of in-class graded programs revealed that some cadets became stymied by a particular error and progressed no further, spending as much as half an hour on the same point. Some of these cadets requested assistance, but a teacher is severely limited in the quality of hints that can be given during a graded event. Sometimes they helped, but often they didn't.

Testing is part of the problem solving process, and identifying and correcting logic and syntax errors are both part of testing. These concepts are explained, but the actual mechanics of interpreting and eliminating errors is often lost on a class. Both intentional and unintentional errors during exercise demonstrations are helpful, but this only scratches the surface of the myriad combinations of errors that a novice student might encounter, and it is impossible to cover everything during our classroom sessions. The course text, being also focused on concepts rather than syntax, provides no help with syntax errors.

The author cannot recall having received any formal instruction on error correcting as an undergraduate student, but computer science majors were expected to figure out these things on their own. This expectation is less realistic in a core course.

2. Background

Research was conducted in three main areas: teaching programming and error correction to cadets, teaching programming to a non-standard audience, and supplemental instruction.

The pervasive theme in teaching programming is that is difficult (particularly in regards to debugging)[2-5]. Programming is not just an academic discipline, it is a multi-tiered skill, requiring students to utilize multiple types of learning simultaneously[6]. Introductory programming curriculums usually follow a standard format, giving a small amount of time to each concept, and then moving on. Like mathematics, the knowledge is cumulative, and if the previous concept is not grasped, the student then has to wrestle with multiple concepts at once.

Usually the focus in an introductory computing course is programming, that is, learning the processes associated with problem solving using a programming language. The language itself is tangential, and yet there is a large amount of overhead associated with learning the syntax of the language being used (This also applies to the compiler and editor or Integrated Development Environment)[7, 8]. Correcting syntax errors is associated with learning the computer language, but is a skill unto itself because syntax error messages are often cryptic. It is not uncommon to see syntax errors that do not address the real problem. For instance, a cadet entered the following code:

```
public class Hello World extends eecs.Gui
{
    public static void main(String[] args)
    {
          printLine("Hello World!");
    } //close main
} //close class
```

This program produced the error `'{' expected`, but the real problem was that the cadet inserted a space in the class name. The cadet had been admonished that identifiers could not include spaces, but novice programmers will have difficulty making a connection between the admonition and the error message.

Another issue here is that we don't make mistakes on purpose. A student may write error-free programs 99% of the time, and encounter her first syntax error on a graded event. Because the student previously had no syntax errors, she had no motivation to learn how to correct syntax errors. Despite the traumatic effect errors have on students, they are ultimately beneficial because the process of correcting them helps students learn more about how the computer works.

Typically, programming is taught to computer science majors. These students have chose computer science as their field of study and it is assumed that they possess more than a passing

interest in the discipline, and thus possess the motivation to tackle the more challenging aspects of the courses. Even then, when students encounter a problem, they fall into two camps, the movers and the stoppers[9]. The movers experiment using the knowledge they have, making controlled modifications in hopes of overcoming the current obstacle. The stoppers throw their hands in the air in frustration and progress no further. Psychologists call this "learned helplessness."[6]

If this is the case for computer science majors, does it also hold true for non-majors? Is there a difference between the two groups? Studies have shown that there is usually a small difference in mathematical aptitude, which is related to a cadet's ability to create algorithms. Non-majors have expressed difficulty in understanding problem specifications, specifically in analyzing them and creating algorithms from them[10].

Different mediums for providing information on error correction were also explored. Because the goal was to provide material that would be available for students outside the classroom, the term supplementary instruction seemed appropriate. However, the term supplementary instruction seems to be associated providing with non-traditional and/or underprepared students[11, 12].

Because technology is the focus of the course, leveraging technology to facilitate this learning made sense. YouTube became the platform of choice. Some research has been done regarding the role of YouTube and education[13]. The fact that anyone can put content on YouTube is both a pro and a con. YouTube contains a wealth of information, but there is no formal peer review, so the information presented is unreliable. Some studies have explored the informal peer review in the form of comments and likes/dislikes.

3. Method

Instructional videos were created using Camtasia Studio 4 and CamStudio 2.0. Both programs were evaluated, and while Camtasia Studio contained more features, CamStudio can produce similar results and has the advantage of being free. Adobe Captivate was considered but was determined to be cost prohibitive.

Video subjects were determined based on observations from previous IT105 semesters. The goal was to tailor instruction specifically to the context in which cadets were making the errors. Methods (subroutines) is the last programming concept taught in IT105, and it created a different context for errors, which the videos addressed. The goal was for all videos to be under three minutes. This seems to be a standard on YouTube, and also takes attention span and frustration levels into consideration. Because cadets were allowed to view the videos in the classroom, the videos contained no audio.

The content of each video starts with a description of the error message. Then a scenario is shown in either RAPTOR or the programming editor in which the error occurs. Messages are shown highlighting each individual element of the error message and the significance. The error is then corrected, with messages showing the steps needed to correct the error.

The videos were introduced to four IT105 classes, containing either 17 or 18 cadets. The introduction was made after the halfway point in the programming phase of class, where cadets are dealing with multiple concepts and errors typically increase. The channel was shown three times, typically at the beginning of class, and cadets were encouraged to favorite/bookmark the site. The students were also referred to the videos when they encountered errors during in-class exercises.

The view count of the videos was checked periodically. The times that the view counts increased suggested that the videos were initially being viewed only by IT105 cadets and principally during in-class programming assignments, but this no longer seems to be the case.

During the last lesson of the semester, a six-question survey was issued to all four classes. 59 out of 71 cadets completed the survey. The final exam took place after the survey was given. There was a significant change in view counts during the final exam, suggesting that cadets took advantage of the resource.

4. Results

4.1 Resource awareness, necessity and consumption

Figure 1 shows the distribution of cadet responses to the question "Did you visit the video channel?" The response "I wasn't aware the videos existed" was added due to a student comment during a program review.

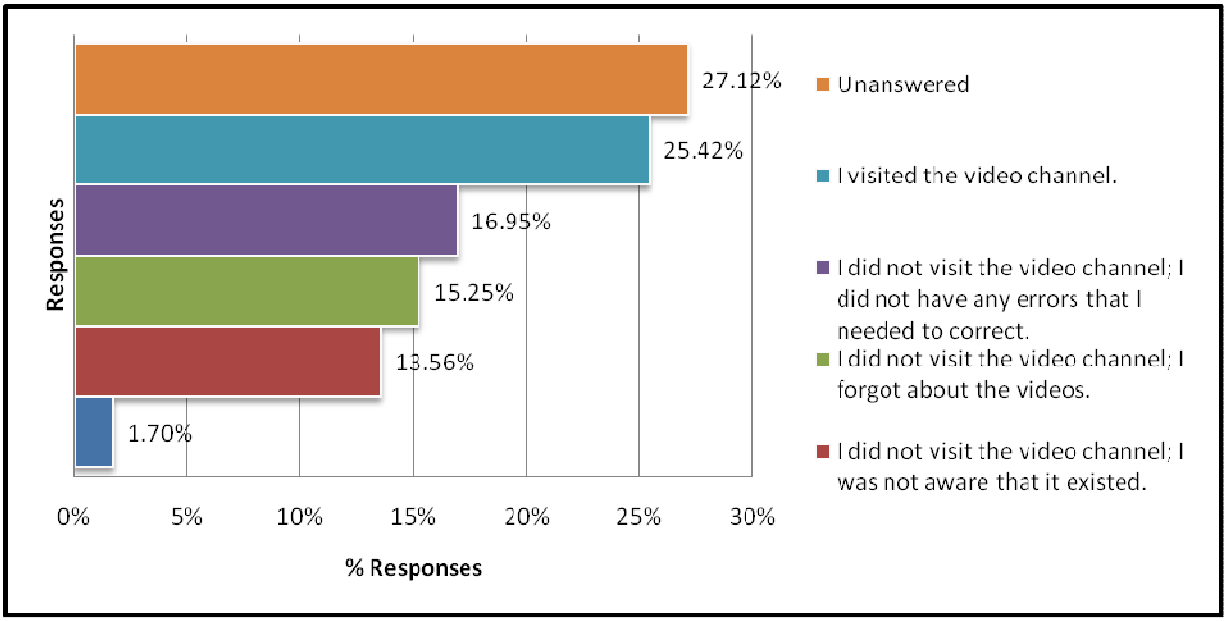

Figure 1: Distribution of resource awareness, necessity and consumption

4.2 Pattern of resource consumption

Figure 2 shows the distribution of responses to the question "When did you visit the video channel?" Cadets were allowed multiple responses on this question. For this reason and because

only one quarter of the responders actually visited the channels, percentages do not add up to 100%.

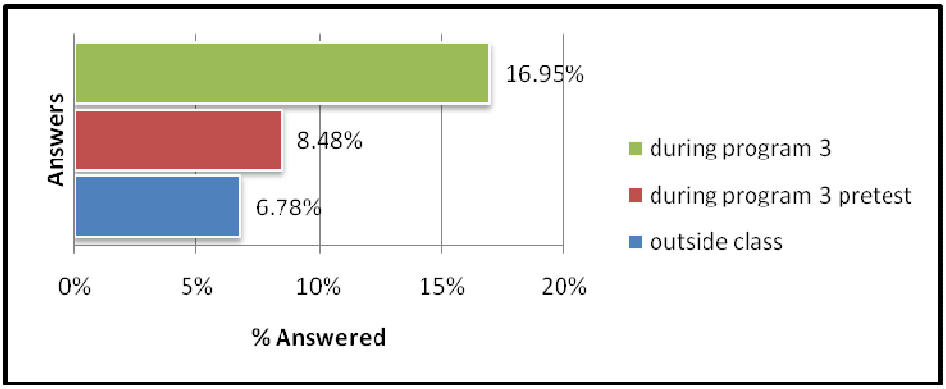

Figure 2: Distribution of resource consumption

4.3 Personal utility

Figure 3 shows cadet responses the question of whether cadets found the videos helpful. The question used a six-level likert item with the following responses: very helpful, helpful, neither helpful nor unhelpful, unhelpful, very unhelpful, non applicable. There were no unhelpful or very unhelpful responses. Non applicable responses totaled 10.169%. The remaining responses were adjusted to a 100% scale.

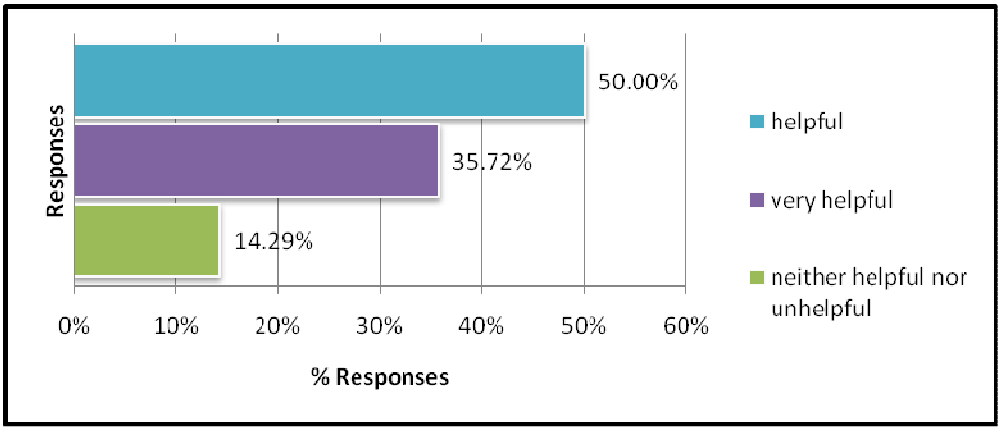

Figure 3: Distribution of personal utility

4.4 General utility

Figure 4 shows cadet responses to the question "Do you think the average IT105 cadet would be interested in content like this?" Cadets were limited to a yes/no response.

4.5 General comments

General comments and suggestions were solicited at the end of the survey. For the most part cadets reiterated issues addressed in the survey. The most prevalent comment is that cadets

wished that more references had been made during class to the videos because they were unaware of the content and struggled during the programming exams with errors that the videos addressed directly. The second most common comment was that cadets did not watch the videos because they either made no syntax errors or they were able to debug their code without assistance, but they felt that the videos were a good resource nonetheless.

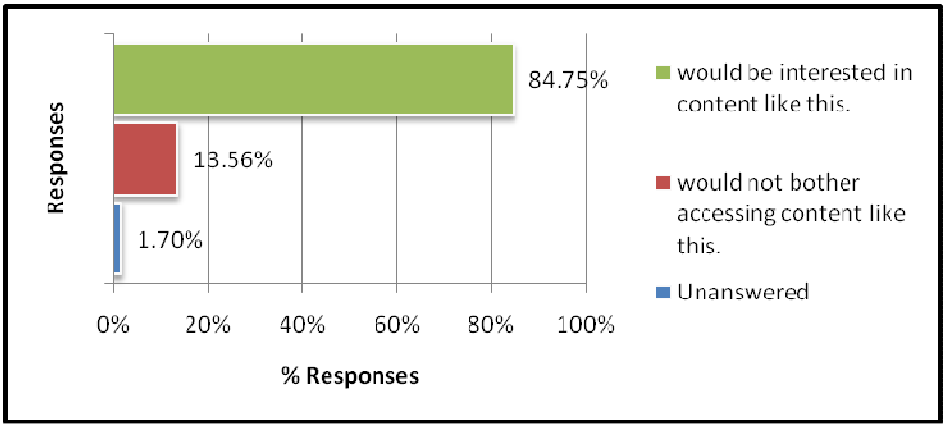

Figure 4: Distribution of general utility

4.6 Grade Improvement

Figure 5 provides a comparison of IT105 grades for three different semesters. The videos were introduced during Fall 2010 between programs 2 and 3. Fall 2010 is the only semester where all sections showed improvement from the program 3 to the final exam, and half the classes showed improvement across all three major programming assignments.

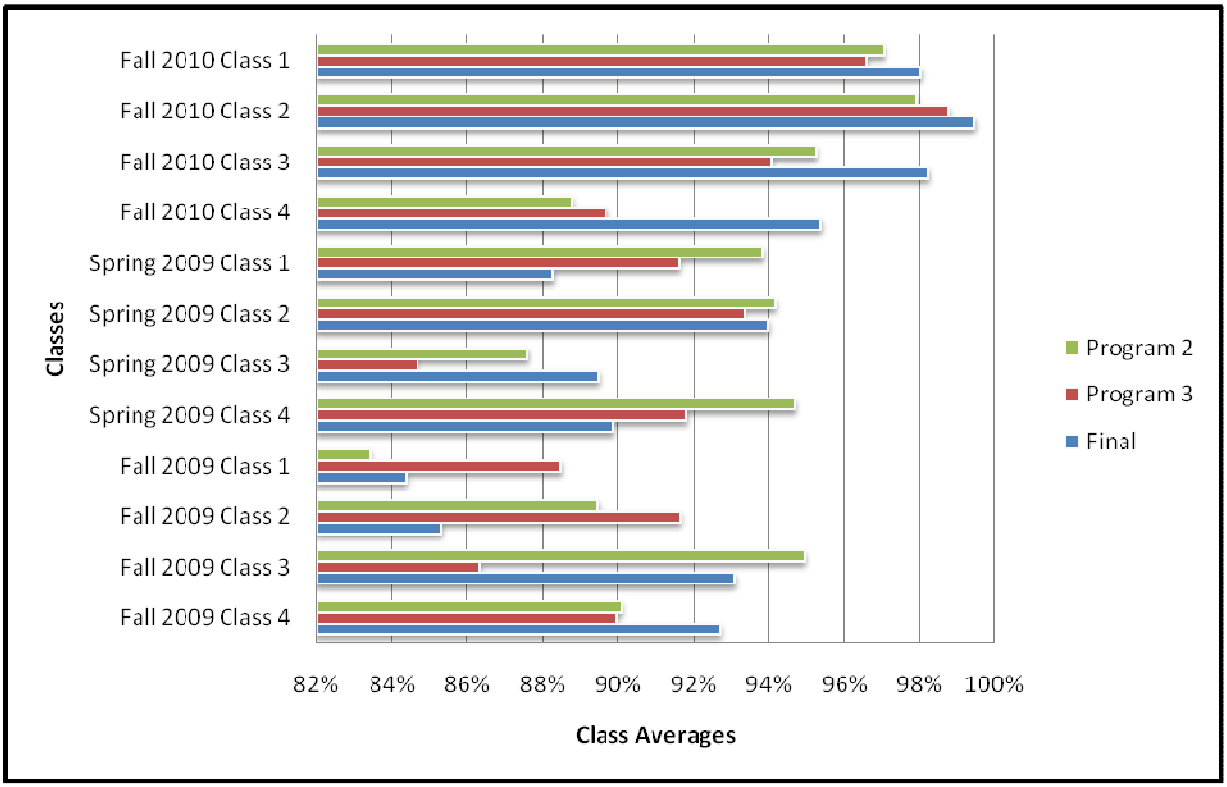

Figure 5: Programming averages across the last three semesters

5. Discussion

Initially, the 25% percent consumption figure was disappointing, but reality is that many cadets do not need this resource. As Figure 5 shows, the worst class average was a B. Only a few cadets really struggle with error correction, so 25% is fitting. The more puzzling question is why did 27% of people that took the survey decline to answer this particular question.

As expected, most cadets only looked at the videos when they were actually working through the labs in the classroom. It was promising to see that some cadets looked at the labs during the non-graded event.

Initially cadet responses to the questions of personal and general utility were met with skepticism. All IT105 cadets are freshmen and some of them are still uncomfortable with providing negative feedback. Comparing grades across all sections helped change my perception. Cadets performed better on the programming portion of the final exam than ever before. The combination of the improvement in grades along with the comments was enough for me to consider the endeavor worthwhile.

The videos also had a favorable impact on teaching. A considerable amount of time was devoted to helping cadets correct errors in code in the classroom during my first semester of IT105—sometimes at the expense of teaching. My rationale was that it was worth it to ensure that cadets gained an adequate understanding of how to debug their code. Some errors seemed important

enough to share with the entire class. Now I am able to direct the cadet to a video when he or she encounters a syntax error; they are free to ask me questions if there is something in the video they didn't understand. This also helps reinforce the concept that the cadet is responsible for his own learning.

6. Summary and Future Work

A significant number of cadets make mistakes while programming, are at a loss at how to correct these mistakes, and reach an impasse. In response videos were created for cadets to view at their leisure that walk them through the process of correcting syntax errors. There was some uncertainty about whether cadets would take advantage of such a resource. This study shows that cadets did watch the videos and found them useful.

The next step is to increase awareness of the videos. Other professors can be implored to mention the videos during their lectures. More videos need to be created. The current videos cover the most common errors, but only a fraction of the errors cadets might encounter. Feedback in the form of direct video comments rather than by survey should also be encouraged.

Bibliography

[1] Martin, C.C., A.W. Terry, W.H. Jeffrey, and M.H. Steven, "RAPTOR: a visual programming environment for teaching algorithmic problem solving", *Proceedings of the 36th SIGCSE technical symposium on Computer science education*, St. Louis, Missouri, USA: ACM, 2005.

[2] Joe, M., F. William, and L. Henry, "Teaching applied computing without programming: a case-based introductory course for general education", *Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education*, Charlotte, North Carolina, United States: ACM, 2001.

[3] Miguel, U., "Teaching and learning computer programming: a survey of student problems, teaching methods, and automated instructional tools": ACM, 1980, pp. 48-64.

[4] Robert, F.M., "Teaching debugging", *Proceedings of the fourth SIGCSE technical symposium on Computer science education*: ACM, 1974.

[5] Yasuhiko, M., K. Kunimi, Y. Setsuo, U. Maomi, and M. Youzou, "A Support System for Teaching Computer Programming Based on the Analysis of Compilation Errors", *Proceedings of the Sixth IEEE International Conference on Advanced Learning Technologies*: IEEE Computer Society, 2006.

[6] Jenkins, T., "On the Difficulty of Learning to Program", *3rd annual Conference of LTSN-ICS*, 2002.

[7] Jenkins, T., "Java with BlueJ or Java and BlueJ", *5th Annual Conference of the LTSN Subject Centre for Information and Computer Sciences*, Newtownabbey, Northern Ireland: Higher Education Academy, 2004.

[8] Raymond, F., and L. Bob, "Teaching programming collaboratively", *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, Caparica, Portugal: ACM, 2005.

[9] Ala-Mutka, K., "Problems in Learning and Teaching Programming", *Codewitz Needs Analysis*: Institute of Software Systems, Tampere University of Technology.

[10] Christine, P., and L. Xiaosong, "Teaching introductory programming to Information Systems and Computing majors: is there a difference?" *Proceedings of the sixth conference on Australasian computing education - Volume 30*, Dunedin, New Zealand: Australian Computer Society, Inc., 2004.

[11] Martin, D.C., and R. Blanc," Video-Based Supplemental Instruction (VSI)", *Journal of Developmental Education* Vol. 24, No. 3, 2001, pp. 12-19.

[12] Painter, S.L., R. Bailey, M. Gilbert, and J. Prior," New Directions for Supplemental Instruction", *New Directions for Teaching and Learning*, 2006.

[13] Marlene, A., D. Teresa, M.M. Eric, T. Cristina, and H. Linda, "Learning from YouTube: an analysis of information literacy in user discourse", *Proceedings of the 2011 iConference*, Seattle, Washington: ACM, 2011.