Covering Programming Language Syntax With Reading Assignments, A Failed Experiment?

Thomas Murphy[1]

Abstract

Introductory Computer Science courses are often taught using object-oriented programming languages such as C++ and Java and computer aided design tools such as Microsoft's Visual Studio and Borland's JBuilder. This can result in extensive course coverage of programming language syntax and software use at the expense of Computer Science concepts. An attempt was made at having students learn programming language syntax through their reading assignments. This was to avoid spending the increasingly large amount of class time covering the volume of syntax associated with object-oriented programming languages.

Introduction

Many students, including engineering and science majors, will only take a single formal course in Computer Science, thus it is important that they are exposed to some of the important concepts in Computer Science (Shackelford, Shackelford and LeBlanc). Spending an appropriate amount of time covering important Computer Science concepts has becomes increasingly difficult when the introductory course involves a complex programming language and a sophisticated software development tool. It is easy for introductory Computer Science courses to become courses in programming language syntax or programming in a specific language.

A brief survey of some commonly used introductory programming texts (Deitel and Deitel 2001, 2002, Liang) illustrates just how much syntax and software use is commonly encountered in introductory programming courses. For example, approximately 20 pages of chapter 1 in (Liang) cover how to create a standard print welcome program using JBuilder and approximately 10 pages of chapter 2 in (Deitel and Deitel, 2002) to do the same using the command line. The mentioned textbooks are popular textbooks for introductory programming classes. They do a good job of covering syntax and have a lot of sample programs.

Since many textbooks do a good job of explaining syntax and provide examples of the use of the syntax, students should be able to learn at least some programming language syntax through a combination of reading their textbook and looking at program examples (this is how many experienced programmers learn new programming languages). An attempt was made at having students learn some of the programming language syntax necessary for an introductory programming course through their reading assignments. This was done to avoid spending the increasingly large amount of class time covering the volume of syntax associated with object-oriented programming languages. The intended benefit was to free up more class time for emphasizing Computer Science concepts.

---

[1]. Associate Professor of Engineering, School of Computing, Armstrong Atlantic State University, 11935 Abercorn Street, Savannah, GA 31419-1997, E-mail: tommurphy@ieee.org

This paper summarizes the experiences of teaching several introductory programming courses, specifically covering programming language syntax during class time versus encouraging students to learn the programming language syntax through their reading assignments.

Goals and Methodology

Previously, the author had covered a significant amount of programming language syntax (C++) in introductory programming courses. This was considered necessary due to the complexity of object-oriented languages such as C++ and JAVA. The primary goal of this work was to move coverage of some of the programming language syntax to reading assignments thus freeing up class time for Computer Science concepts such as: algorithm development, problem solving, program development, and code modification and reuse. A secondary goal was to illustrate to students that they could learn technical material on their own through reading the textbook. Reading and understanding technical material from books/journals is an essential part of life-long learning.

Formally having students learn programming language syntax via reading assignments was initially tried in an introductory C++ programming class during the spring of 2001. The course is 4 credit hours, meeting 3 hours a week for lecture and 2 hours a week for lab. The class was predominantly made up of freshman and sophomore Computer Science and Computer/Electrical Engineering majors. The course was taught using C++, Microsoft's Visual Studio, and Ford and Tropp (Ford and Trop 1999) as the textbook. At the time, the Deitel and Deitel textbook (Deitel, 2001) was used in the second programming course. Students were strongly encouraged to also purchase this book. In subsequent semesters the Deitel and Deitel textbook (Deitel, 2001) was used for this course.

The class was clearly told at the start of the semester (and periodically reminded when it appeared they were not reading) that much of the syntax details would not be covered in class and that they were responsible for learning this through reading the text. The course syllabus given to students at the beginning of the semester and available via the course web site contained a detailed weekly schedule of topics and reading assignments. In addition, at the beginning of each week daily topics and reading assignments were posted on the course web site. This was considered necessary to remind students of their reading assignments and to account for times when the course did not exactly follow the outline in the syllabus.

Since less syntax was to be covered in class, a way of encouraging textbook reading to learn the syntax was necessary. Several good methods of encouraging textbook reading without sacrificing student homework effort are suggested in (Gray). Briefly, the reading summary (RS) method involves having a randomly selected team of students summarize the days reading assignment at the beginning of class. The problem summary (PS) method involves a randomly selected team of students summarize one of the homework problems. The combination of these was used by Gray to encourage textbook reading. His classes consisted of 4- 6 teams of 2-3 students each. The courses ranged from freshman to senior level. As many introductory programming courses have a much higher enrollment than this (my courses are usually 25 – 30 students) and do not use teams an alternative to the RS method was sought.

Two methods of encouragement/punishment were used to encourage students to complete the reading assignments. It was hoped that a combination of unannounced quizzes and in class assignments would be enough to encourage students to complete the reading assignments. Students were told at the beginning of the semester (and periodically reminded) that they would have unannounced quizzes and class work covering their reading assignments. The quizzes and class work were designed to be fairly easy if the reading assignment had been performed and as the case in (Gray) were made a significant percentage of the final grade (20%). The quizzes/work were given approximately once a week.

A variation of the problem summary (PS) method was already being used in my programming courses and is a fairly common teaching method in programming courses. Specifically project solutions and other examples were discussed in detail usually by the instructor but occasionally by students. This was done to illustrate design process, algorithm and program development, and as a side benefit some syntax was covered. In addition, some of the more complex C++ syntax, such as creating objects, was specifically covered in class.

Expectations and Results

It was expected that there would be initial resistance to the idea of learning programming language syntax through reading assignments, specifically that this would be perceived as extra worked required by the instructor of this particular section. Students often expect that everything they should learn will be specifically covered in class. Class time was specifically set aside at the beginning of the semester for catch up so as to leave time to cover some syntax if it became necessary. Once students got used to the idea and realized that a significant portion of their grade depended upon quizzes/work based on the reading, it was expected that they would perform the reading assignments more diligently and the catch up time could be used for additional programming examples.

Assessment of whether the reading assignments were being completed was accomplished via the quizzes/class work (served dual use as both encouragement and assessment), through instructor observations, and through a written survey given during the last week of class. The survey can be found in the Appendix. In addition, some anecdotal information of whether the reading was completed is discussed.

The sections quiz/class work average was 5.4 out of 10 with only one student achieving a quiz average above 9/10 and the next highest being just above 8/10. Recall, the quizzes and class work were designed to be fairly easy if the reading assignment was done. I had expected an average of around 8/10. For comparison, the class's project average and lab average were both around 7.5/10 and nine students had project averages exceeding 9/10. From this, one can infer that the students did not take the quizzes and thus the reading assignments very seriously.

The results of the written surveys were surprising. 16 out of 20 students completed the survey (7 dropped before the survey was given). The self reported GPA of these students was 2.9/4.0 and 14 of the students were either Computer Science or Engineering majors. When asked if they performed the reading assignments, 6 answered yes and 11 answered no (rarely, barely etc were included in nos). When asked to list up to three of the most helpful things in learning course material, lab work was mentioned most often followed by posted program examples. Class lectures and projects tied for third. Specific ratings (averages) of helpfulness of various course materials (from 0 to 3 with 3 being most helpful) are summarized in Table 1 (column labeled reading section).

Another section of the same course taught by a different instructor was used for comparison. This section covered the same topics in a similar manner except that more syntax was explicitly covered during class time and reading assignments were not as strongly emphasized. Similar projects and exams were given, and the final grade distributions of both sections were similar. 17 out of 24 students completed the survey (7 dropped before the survey was given). The self reported GPA of these students was 3.2/4.0 and 12 of the students were either Computer Science or Engineering majors. When asked if they performed the reading assignments, 6 answered yes, 8 answered no, and 3 did not answer. When asked to list up to three of the most helpful things in learning course material, class lectures was mentioned most, followed by lab work, and going over solutions. Specific ratings (averages) of helpfulness of various course materials (from 0 to 3 with 3 being most helpful) are summarized in Table 1 (column labeled comparison section).

Course Material	Reading section	Comparison section
Class Lectures	2.2	2.7
Reading Assignments	0.9	1.1
Posted Program Examples	2.6	2.3
Lab Work	2.8	2.7
Projects/Homework	2.6	2.5
Quizzes	1.9	1.4
Exams	2.0	1.7
Going over solutions	2.7	2.7

Table 1: Average Ratings of Helpfulness of Course Material

It was expected that students would consider class lectures, posted program examples, lab work, and projects the most helpful. Quizzes and exams were expected to be the least helpful. From that standpoint the results were as expected. However the very poor rating of reading assignments was unexpected especially for the section where reading was strongly emphasized.

Before the semester started, I expected that emphasizing the reading assignments would lead to more students performing them and more students considering reading their textbook helpful in learning the course material. However, the low percentage of students performing the reading assignments and the low helpfulness rating was not entirely unexpected. As the semester progressed, anecdotal evidence suggested such. It became apparent before midterm, that students were not satisfactorily performing the reading assignments. Quiz grades were very poor, in class work was being done poorly and taking too long, and I was getting questions on program example syntax that made it apparent that most students were not performing the reading. In addition, during the lab many students had problems identifying errors from the compiler messages and line numbers. Most errors in introductory classes are syntax errors. The class was reminded of the importance of syntax and the reading assignments, but things did not improve much by the end of the semester.

Even more unexpected was which students said they did the reading. The expected grade of students was compared to whether or not they did the reading assignments. The combined results of the two sections was: only 1/7 who expected an A did the reading, 3/8 who expected a B did the reading, 4/9 who expected a C did the reading, and 2/3 who expected a D did the reading. No student expected to get an F and one student who did not do the reading expected to get a B or C. The self reported GPA of students who did the reading versus students who did not was about the same 3.1/4.0 versus 3.0/4.0.

Conclusions and Other Attempts

Overall, the attempt at covering programming language syntax using reading assignments was disappointing. By midterm, anecdotal evidence suggested that students were not satisfactorily

performing the reading assignments. The final written survey and quiz averages confirmed this. Thus it ended up being necessary to go back to covering more of the programming syntax during class.

A better method of encouragement/punishment is necessary for this idea to work in the future. Using unannounced quizzes as the incentive failed and even though I was initially reluctant to attempt Gray's RS method, the embarrassment factor during a poor oral presentation may be the necessary incentive. The reluctance of using the RS method was due to several factors: 5-8 minute summaries using up 10%+ of a fifty minute class, student tardiness and absences, and some aversion to setting up teams just for this purpose. I seldom use a team approach for work in introductory programming classes since encountering several cases of a "good" student pulling his "poor" friend along to upper division classes.

In later courses, and more informally an alternative approach was tried. The detailed reading assignments were continued, some syntax was covered during class, but I started using the lab period to both introduce and reinforce the programming language syntax. Working examples illustrating the syntax for the concepts covered that week were provided to students. Students were required to run these and then make minor to substantial modifications (play with the code) depending upon the topic. The instructor and a lab assistant were available to answer any questions (short of writing and typing the modified code).

This approach worked much better and worked especially well in a summer JAVA course taught to predominantly Information Technology and Information Systems majors. The summer course was a short semester, so the course met 2 hours every day. 30 - 45 minutes were used for lecture followed by the rest of the time doing programming work in the lab. The hands on use of the language and compiler with the instructor/assistant as backup worked well for most students. There were several students who relied too much on their neighbor and required an overly large percentage of instructor/assistant time. Some experiments even though designed to be straightforward and short often took average students 1 to 1.5 hours. And for the complex syntax of GUIs and objects even good students had difficulty finishing. Even using this approach, it was very difficult to get students to use their textbooks. Most lab projects cited specific pages in the textbook, but it was apparent that few students read the background material even during the lab period.

Using a lab period to introduce/reinforce syntax allowed more class time to be used for Computer Science concepts. However, this requires lab space, extensive lab time, well thought out lab assignments, and good help. The JAVA course would not have worked nearly as well without the excellent teaching assistant I had.

References

Deitel, H.M. and Deitel P.J. (2001) *C++ How to Program, third edition*, Prentice-Hall, Upper Saddle River, New Jersey.

Deitel, H.M. and Deitel P.J. (2002) *JAVA How to Program, fourth edition*, Prentice-Hall, Upper Saddle River, New Jersey.

Ford, William H. and Tropp, William R. (1999) *Introduction to Computing using C++ and Object technology* Prentice-Hall, Upper Saddle River, New Jersey.

Gray, Robert I. (2001) "A Solution to the I-Never-Do-the-Text-Reading-Assignment Problem," Proceedings of the 2001 ASEE Southeastern Section Conference, April 1-3, 2001, Charleston, SC.

Liang, Y. Daniel (2002) *Introduction to JAVA Programming with JBuilder 4/5/6, second edition*, Prentice-Hall, Upper Saddle River, New Jersey.

Shackelford, Russell L. (1998) *Introduction to Computing and Algorithms*, Addison-Wesley, Reading, Massachusetts.

Shackelford, R. L. and LeBlanc, R.J. (2001) "Introducing Computer Science Fundamentals Before," Proceedings of the 27th Frontiers in Education Conference, Volume 1, Pages 285 – 289, 1997.

Appendix

CSCI 1301 Course Survey

Rate your level of knowledge of the following topics before and after completing CSCI 1301. (1 indicating little or no knowledge of the topic and 3 indicating good knowledge of the topic)

Topic	Before CSCI 1301 Little		Good	After CSCI 1301 Little		Good
Program Format	1	2	3	1	2	3
Conditional Structures (if, switch)	1	2	3	1	2	3
Iterative structures (for, while)	1	2	3	1	2	3
Functions	1	2	3	1	2	3
Classes and objects	1	2	3	1	2	3

Rate the helpfulness of each of the following in learning the course material. (0 indicating not applicable, 1 indicating little or no help and 3 indicating very helpful)

	Not App	Little help		Very helpful
Class lectures	0	1	2	3
Reading Assignments	0	1	2	3
Posted program examples	0	1	2	3
Lab work	0	1	2	3
Projects/Homework	0	1	2	3
Quizzes	0	1	2	3

Exams	0	1	2	3
Going over solutions to labs, projects, quizzes, and exams	0	1	2	3

List up to three things from above that were most helpful in learning the course material (1 being the most helpful).

1.

2.

3.

What is your major?

What is your overall GPA (out of 4.0)?

What grade do you expect to receive in this course (A, B, C, D, F)?

Did you do the reading assignments (yes or no)?

Additional Comments/Suggestions may be written on the back.

.

Thomas Murphy

Thomas Murphy is an Associate Professor of Engineering at Armstrong Atlantic State University. Prior to this he taught in the Department of Mathematics and Computer Science Department at Georgia Southern University and in the Department of Electrical and Computer Engineering at The Citadel. He received his PhD and M.E. in Electrical and Computer Engineering from the University of Florida and his B.S. in Electrical Engineering from the University of Notre Dame. His interests are primarily in the areas of signal processing, digital logic, control systems, and computer applications in these areas.