Viope as a Tool for Teaching Introductory Programming: An Empirical Investigation

Jeffrey Carver and Lisa Henderson
Department of Computer Science and Engineering
Mississippi State University
{carver, lisah}@cse.msstate.edu

Abstract

In this paper we describe the use of a tool from Viope for teaching introductory programming. We have noticed in our previous courses that the students often have trouble connecting the small classroom exercises with the larger laboratory projects. This tool allows the students to get extra practice with those concepts to help ensure they are understood. In this study data was collected using a survey. We surveyed students at the end of a semester in which the tool was not used to gather information about where the tool might be useful. Then we had students in a second semester use the tool and complete a similar survey. The results of our study showed that while there were some consistent complaints about the tool, overall the students found it useful enough to indicate they would like to use something similar in later semesters.

1. Introduction

Introductory programming courses at universities are focused on teaching inexperienced students how to write software programs. This task is a challenging and necessary one if these students are to be successful in their careers as software engineering students. Over our 5 years of experience in teaching introductory programming courses we have found that there are a series of topics that students commonly have difficulty learning. We believe that much of that difficulty can be attributed to the lack of hands-on experience the students have with those topics.

In our typical introductory programming course, the students learn the material from the instructor during the lecture. They may perform some small classroom exercises to reinforce that material. Then the students participate in mandatory laboratory sessions in which they apply the concepts they have learned in class to develop larger programs. Finally, the students must show their understanding of the course material on exams. A student's grade in the course is based on a combination of their laboratory scores and their exam scores.

It has become evident through observing the behavior and performance of the students that some students require additional practice to fully grasp the complicated programming concepts that are covered in these introductory courses. The students have problems applying the small examples from class to the larger laboratory assignments. Some students are motivated to work practice programming problems on their own to ensure they understand the concepts, but often students will not know what types of practice problems to work or will not have access to good examples to use to develop their skills and understanding.

This paper describes a potential solution to the above problem along with some initial empirical results. In section 2, we first describe the related work in this area. Section 3

Proceedings of the 19th Conference on Software Engineering Education & Training (CSEET'06)
1093-0175/06 $20.00 © 2006 IEEE

describes a tool from Viope that is aimed at aiding the teaching of introductory programming, which has been made available to us for use in our courses. In section 4 we describe the methodology we used to conduct a series of surveys to gather information about the tool and its use. In section 5 we draw some conclusions based on the data. Finally we conclude the paper with a summary and with future research directions in section 6.

2. Related Work

There are various types of tools to facilitate learning how to program. Rongas, et. al. classify tools for learning introductory programming into four categories based on their function. The first category is the Integrated Development Environment (IDE), which contains a set of programming tools (e.g. editor, compiler, debugger) within one application. The second category contains Visualization tools that let students see what happens during the execution of a program. The third category consists of Virtual Learning Environments. These tools encapsulate the set of tools needed for one specific programming course. Finally there are Submission systems that facilitate submission and evaluation of programming assignments [4]. The tool that we use here, described in Section 3, is most similar to a Submission system.

Researchers in other areas of the software lifecycle have recognized the both the need and the benefits of using tools to facilitate education. In studying the most effective methods for teaching modeling languages, Alfred, et. al. found that commercial modeling tools were not adequate for use in educational settings. They created a tool and then solicited feedback from the students on the benefits the tool provided in their classroom assignments. Most students found some benefit from using the tool [1].

In the realm of teaching programming there is some precedent for using tools. Lim et. al. describe a basic programming tool and two extensions to that tool. The basic tool evaluates the correctness of code based on a library of examples. The authors created two add-ons for the tool: a program slicing tool to aid in debugging and a text generator to explain the algorithm in plan English. In an initial evaluation of these add-ons, the results were quite favorable [3]. This study showed that tool support can be beneficial to learning how to program. Another group of researchers developed ActiveTutor, a tool focused on teaching object-oriented programming. The language-independent tool animates algorithms to help students understand how they work. So, rather than teaching how to program, the tool teaches students important underlying concepts about the whole family of object-oriented languages. A study of this tool showed that teaching the concepts of object-oriented programming can be aided by tool support [2].

There are two additional tools that have a more similar goal as the tool we discuss in Section 3, CodeLab tool from Turing's Craft (www.turingscraft.com) and MyCodeMate (www.MyCodeMate.com). These tools both allow students to enter code into the tool and then provide feedback on the correctness of that code. Each tool has a slightly different approach and provides different types of feedback. For example, MyCodeMate gives the students hints on completing the programs but does not check for logic errors in the code written by the students. While both of these tools have positive recommendations from their users, we did not find a rigorous empirical evaluation of either tool.

3. Viope Tool

In this study we evaluated a tool from Viope Solutions (www.viope.com) that aids in teaching introductory programming. (From now on referred to as "the Viope tool".) This tool has many features ranging from content modules, that can be used to actually teach course material, to programming problems that students can use to ensure that they grasp the course

material. In our study, we evaluated only the programming problems. We made use of these problems as a mechanism for the students to practice and understand the material that was taught in class. We also intended for these problems to help the students prepare for the exams.

The portion of the Viope tool that we investigated provides the students with a description of a problem that must be solved and gives a specification for the output of the practice program. These practice exercises are rather short (e.g. under 50 LOC). The shortness of the programs allows the Viope tool user to enter the code directly into the tool, rather than have to use an external editor. Once the code has been entered, the Viope tool will compile and, if successful, execute the code. The user is then provided with feedback both about compilation failures (syntax errors) as well as runtime failures (logic errors). To determine the runtime failures, the Viope tool attempts to match the output of the user's program with the expected output. If the outputs do not match, the user is informed of the discrepancies and then can debug and try again.

We hypothesized that the Viope tool would be beneficial to the students by providing extra practice problems to apply the class material on a small scale prior to working on more difficult laboratory assignments. These exercises, along with interactive feedback from the Viope tool about the correctness of their solutions, should help the students better understand any mistakes they make and correct those mistakes.

4. The Empirical Study

Our ultimate plan is to conduct a controlled study to analyze the benefits of the tool in detail. But, prior to conducting such a controlled and expensive study, it is a good practice to first validate the ideas on a smaller scale [6]. This initial study was meant to judge the feasibility of using the Viope tool to help ensure that a more controlled study in the future would be productive. It was important for us to ensure that the students were seeing some benefit from the Viope tool. Therefore, the goal of this study was to understand a) whether there is a need for the tool and b) whether the tool actually provides some benefit in the areas promised. To accomplish this goal we collected information from two groups of students, one who used the tool and one who did not use the tool.

4.1 Setting

This study was conducted in during two semesters of CSE 1284 (Introduction to Computer Programming) at Mississippi State University (Spring 2005 and Summer 2005). The focus of this course is "Introductory problem solving and computer programming using object-oriented techniques. Theoretical and practical aspects of programming and problem solving."[1] This course covers the following topics: I/O and file I/O, arithmetic expressions, selection structures, iteration, functions, strings, arrays, classes, and pointers.

In the Spring 2005 semester, there were 68 students in two sections of the course (27 in one section and 41 in the other section), and these students did not use the tool. These students make up the control group for our study. In the Summer 2005 semester, there were 14 students in one section of the course, and these students did use the tool. These students make up the experimental group for our study as described in Section 4.2.

[1] Taken from the catalog description of the course.

4.2 Research Methodology

When the tool was made available to us (mid-way through the Spring 2005 semester), we examined the tool and supporting materials to determine the set of activities in which the tool should be beneficial. Based on our understanding, we believed that the tool would:

1. Reduce the time the instructor and TA spent during office hours working with the students to understand the concepts covered in class,
2. Help the students better understand some of the more difficult concepts covered during the course, and
3. Provide the students with material to aid in their study and test preparation.

To understand whether the tool helped students in these areas, we used the qualitative data collection mechanism of a survey to collect data about these topics. A survey is a tool commonly used in software engineering research for gathering both qualitative and quantitative data from a large subject pool [5]. Based on these three items, we created a survey that was distributed at the end of the Spring 2005 semester. The results from this survey provided data that we used to understand how the students without the tool behaved. At the end of the Summer 2005 semester, in which the students used the tool, we also gave them a survey. In this survey we reused all of the questions from the Spring 2005 survey and added a few questions to specifically evaluate the effectiveness and usefulness of the tool. The responses from these two surveys make up the data that we used in Figure 2 and Figure 3 to gain some understanding of the need for the Viope tool. Figure 1 provides an overview of the study design.

Semester	Students	Use of tool	Focus of Survey
Spring 2005	68	No	• Programming Effort • Expected Grade • Additional questions
Summer 2005	14	Yes	• Programming Effort • Expected Grade • Additional Questions • Use of the tool

Figure 1 – Study Design

5. Results

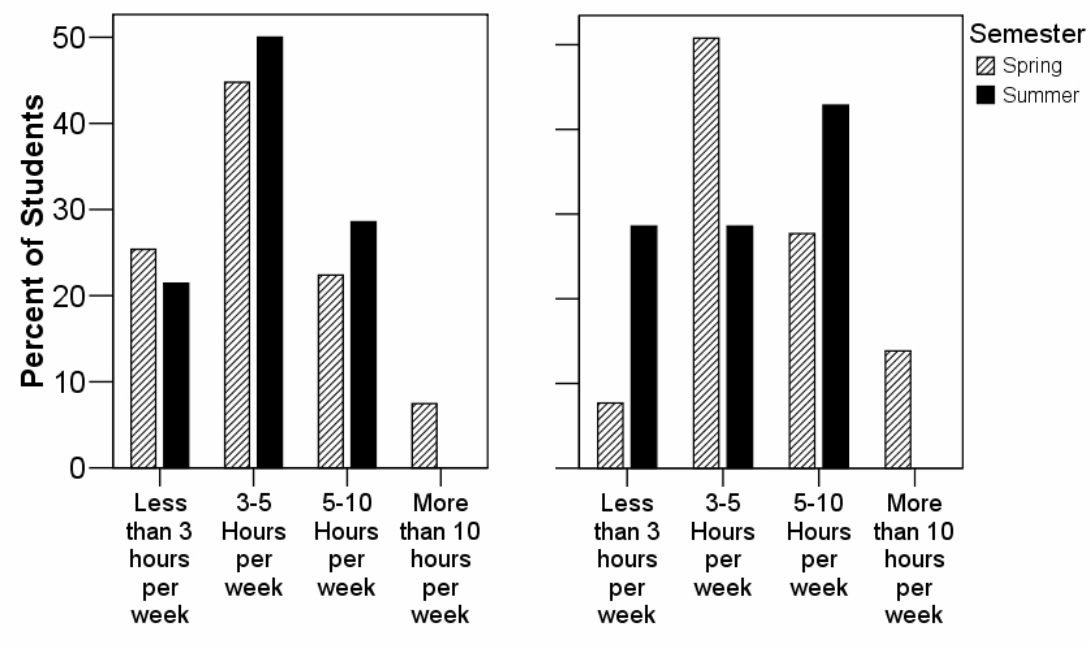
The surveys provided a set of metrics to gauge the usefulness and effectiveness of the tool. Space constraints prohibit us from presenting an analysis of all of the metrics, so we focused on a few of them. In Section 5.1 we compare the responses from students in the Spring semester (no tool) to the responses from the students in the Summer semester (tool) to evaluate any impacts of the tool. We focused on the amount of effort the students spent working on their programming assignments and the students' expected grades. In Section 5.2 we analyze the opinions of the students who used the Viope tool.

5.1 Comparison between Spring and Summer

Keep in mind that the goal of this study was to measure the potential usefulness of the new tool and not to conduct a strict statistical evaluation of the tool. The first metric we examined was the amount of time the students reported working on their programming assignments. Remember that one of our earlier observations was that the students often had trouble

transferring the concepts from small classroom exercises to their larger laboratory assignments. Our belief was that if the tool was really helping them to better understand the fundamental concepts and how to apply them, then the students should spend less time working on their laboratory programming assignments.

In the two surveys we asked the students to report the amount of time they spent working on programming assignments before the midterm and after the midterm. The reason for splitting the semester in half was because the types of programs assigned after the midterm were more difficult than those assigned before the midterm. So, we believed that this distinction would be useful. The options were: 'Less than 3 hours per week', '3-5 hours per week','5-10 hours per week', and 'More than 10 hours per week'. Figure 2 shows that in the Summer semester (using the Viope tool) the students tended to spend less time working on programming assignments than in the Spring. This result is especially true after the midterm when the material covered became more difficult. A large percentage of the students from the Summer semester (~30%) averaged less than 3 hours while less than 10% of the Spring students averaged less than 3 hours. Furthermore, about 15% of the Spring students spent more than 10 hours per week while none of the Summer students spent that long. This result suggests that something that occurred during the Summer semester allowed the students to complete their programming assignments with less effort. We will discuss some potential alternative explanations in Section 6.



(a) Before Midterm (b) After Midterm

Figure 2 – Programming Effort

Another variable that we analyzed was the student's expected grade in the course. The expected grade is a subjective measure of the performance of the students in the course. Because the survey was given prior to the final exam, the student's estimate of their grade reflects their believed understanding of the material and how they would perform on the final exam. Figure 3 shows the distribution of expected grades between the two semesters. As can be observed, the students in the Summer semester (who used the tool) estimated a higher percentage of A's than the Spring semester. Furthermore, no one in the Summer expected to

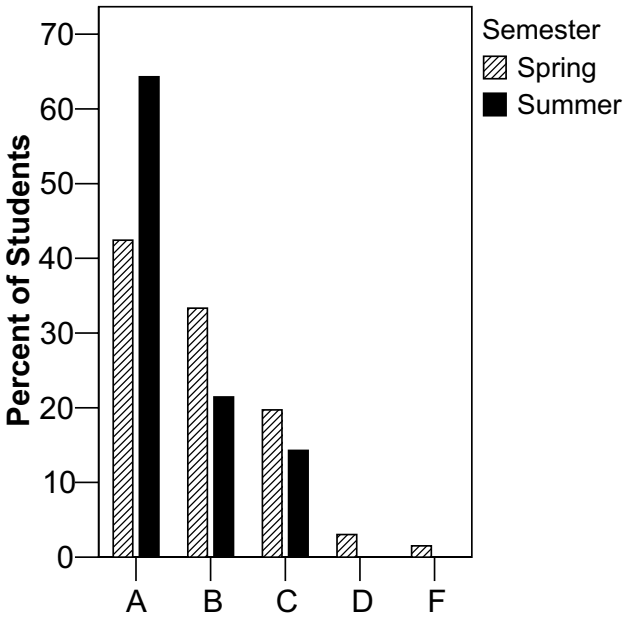


Figure 3 – Expected Grades

receive a D or F while there was a small percentage in the Spring that did expect these low grades.

5.2 Evaluation of the tool

In addition to collecting data to compare the two semesters, we asked the students who used the Viope tool to provide feedback on their experiences. We focused on three questions.

1) How would you rate the Viope tool?
2) Did the Viope tool help you better understand the course material?

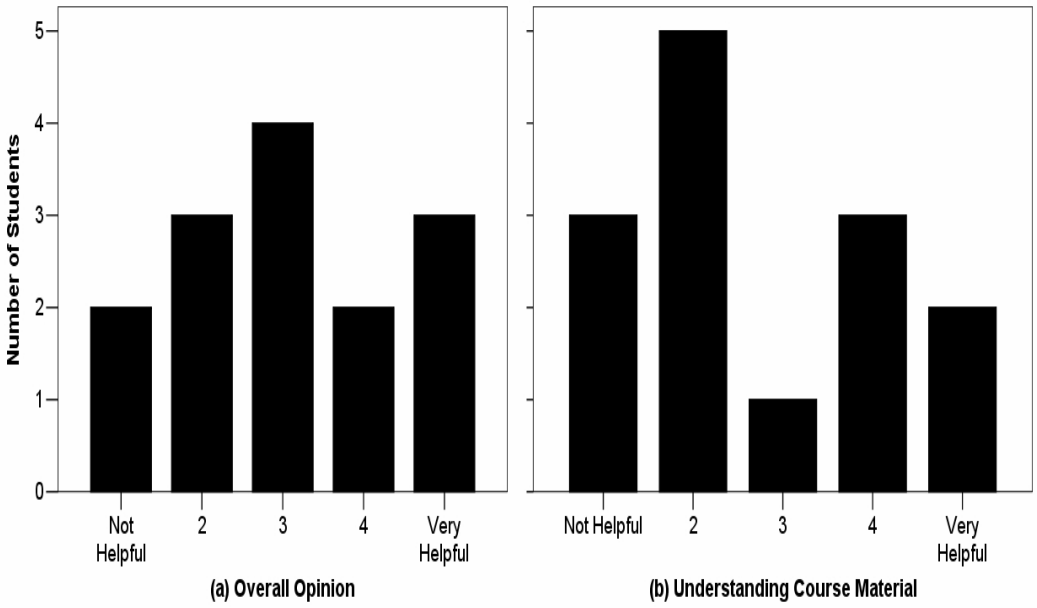


Figure 4 – Evaluation of the Tool

3) Would you like to use a similar tool in subsequent programming courses?

The responses to these three questions appear to be contradictory. In Figure 4 part (a) shows that the students overall opinion of the tool's helpfulness has approximately a normal distribution, while part (b) shows that more students tended to find the tool less helpful in understanding the course material. These results seem contradictory to the results in Figure 5 that overwhelmingly (65% to 35%) show the students would like to use a similar tool in their next programming course.

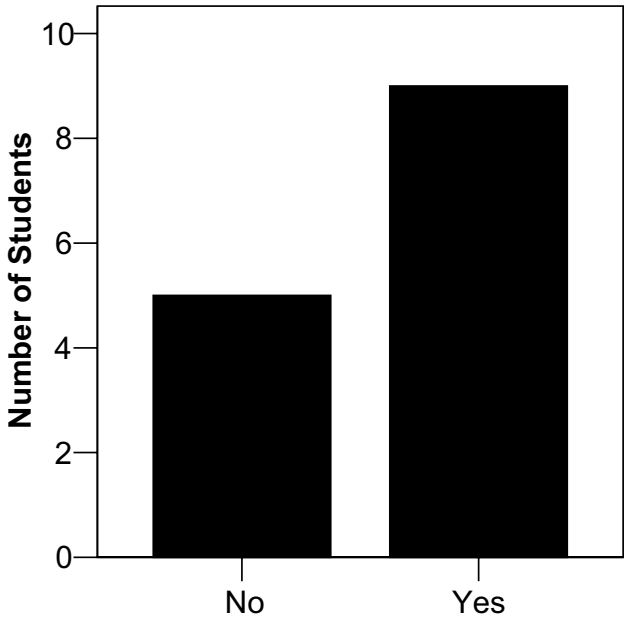


Figure 5 – Use tool again?

In addition to asking questions that required the students to respond based on a predetermined scale, we also collected some qualitative data by providing the students a place to give feedback on the tool. This information helped us better understand the discrepancy in the responses to the three questions. When examining the responses from those who rated the tool as being less helpful it appears that the main concern has to do with the mechanics of the tool. For example, in order to successfully complete a programming problem, everything in the output must match exactly with the Viope tool's expected output. Many students complained that this requirement was too restrictive and prevented the tool from being as helpful as it might have otherwise been. This issue was especially evident in one of the exercises where the output was phrased in way that was different from what the student's would normally have expected.

6. Conclusions and Future Work

As with any study, especially an exploratory and relatively uncontrolled survey-based study as this one, there are some threats to the validity of the results that must be taken into account when drawing conclusions. The main threat to validity we face in this study is that the students who were surveyed came from two different semesters. It is possible that the students who enrolled in the Summer semester were a different population than those who enrolled in the normal Spring semester. This fact must be taken into account when interpreting the results presented in Section 5.

Based on the results, it appears the tool may provide some benefit. We saw that the average amount of time spent programming was less in the Summer than in the Spring. We also saw that overall the expected grades of the students in the Summer was higher than in the Spring. These results could be explained by the difference in semester as described above. But, at least a portion of this result is likely attributable to the use of the tool. While there was a wide distribution of opinion concerning the helpfulness of the tool, a large majority of the students who used the tool indicated that they would like to use a similar tool in future courses. The students did have some doubts about some of the specific peculiarities of the tool, but we can conclude that the basic idea was seen as being helpful.

Our original goal for this study was to determine whether there was enough evidence to expose the tool to a more controlled study. Based on these results, we have decided to perform such a study. Currently, we are evaluating the tool by having one section of a course use the tool while another section does not use the tool. By having both the treatment and control group in the same semester, we hope to reduce the main threat to validity in the study. To do this evaluation we are collecting two types of data. First, the instructor of the course is keeping track of how much time students from each section visit during office hours. Second, the instructor is keeping track of the amount of email questions coming from students in each section. This data will help us to better understand whether the tool helped the students understand the material.

7. Acknowledgements

We would like to thank Viope for making the tool available to us free of charge to perform this evaluation. We would also like to thank the students of CSE 1284 at Mississippi State University in the Spring and Summer 2005 semesters for participating as subjects in our study. Finally, we would like to thank the reviewers for their insightful and helpful comments.

8. References

[1] Alfert, K., Pleumann, J., and Schroder, J. "Software Engineering Education Needs Adequate Modeling Tools". in *Proceedings of the 17th Conference on Software Engineering Education and Training, 2004.* . 2004.

[2] Fournier, J.P. "Activetutor". in *Fifth IEEE International Conference on Advanced Learning Technologies, 2005. ICALT 2005.* . 2005.

[3] Lim, N.R.T., et al. "Ansi C Program Slicing Tool and Text Generator for an Interactive Learning Environment". in *Fifth IEEE International Conference on Advanced Learning Technologies, 2005. ICALT 2005.* . 2005.

[4] Rongas, T., Kaarna, A., and Kalviainen, H. "Classification of Computerized Learning Tools for Introductory Programming Courses: Learning Approach". in *Proceedings of the IEEE International Conference on Advanced Learning Technologies, 2004.* . 2004.

[5] Seaman, C.B., *Qualitative Methods in Empirical Studies of Software Engineering.* Software Engineering, IEEE Transactions on, 1999. **25**(4): p. 557-572.

[6] Shull, F., Carver, J., and Travassos, G. "An Empirical Methodology for Introducing Software Processes". in *Joint 8th European Software Engineering Conference (ESEC) and 9th ACM SIGSOFT Foundations of Software Engineering (FSE-9)*. 2001. Vienna, Austria.