Using Lab Practica to Evaluate Programming Ability

A.T. Chamillard
Department of Computer Science
U.S. Air Force Academy, CO 80840
703-767-6338
achamillard@hq.dcma.mil

Jay K. Joiner
Stone Analytics, Inc.
San Diego, CA 92123
858-503-7540
jkj@stics.com

Abstract

One of the largest challenges facing educators teaching courses with a significant programming component is deciding how to evaluate each student's programming ability. In this paper we discuss how we have addressed this challenge in an introductory computer science course and statistically analyze the results to examine potential inequities in our approach.

Keywords

Programming ability, programming evaluation, introductory computer science.

1 Introduction

There are obviously numerous approaches for teaching programming and evaluating programming ability. Each of these approaches has benefits and drawbacks, so it is reasonable to combine the approaches in a course that contains a non-trivial amount of programming [1]. Researchers have found that collaborative learning, where students are encouraged to discuss concepts and implementation details with other students during the course of a project, can be an effective technique [3]. But how do we then evaluate an individual's ability from a project they created in collaboration with others? This paper describes an approach using lab practica to measure individual programming ability, discusses numerous administrative issues associated with that approach, and demonstrates, through statistical analysis, the apparent equity of our implementation of the approach.

All students attending the U.S. Air Force Academy are required to take an introductory course in computer science (CompSci 110). The key topic in this course is problem solving with computers, so we start by helping the students develop their problem solving skills. Students then learn how to use these skills to solve problems using computers by constructing programs using the Ada programming language.

SIGCSE 2001 2/01 Charlotte, NC, USA
© 2001 ACM ISBN 1-58113-329-4/01/0002...$5.00

The next section describes the lab practica we have incorporated into the course to measure individual programming skills, while Section 3 presents details and issues associated with the administration of the practica. Section 4 provides statistical analysis of student performance on the practica to evaluate the equity of our implementation of this approach. The final section presents our conclusions and comments on our incorporation of lab practica into some of our other courses.

2 Lab Practica Description

A lab practicum is an in-class lab that the students are required to complete within a set period of time (90 minutes). Students must develop and test a complete program solving a problem that they are presented with at the beginning of the time period. They are allowed to use a handout containing syntax for all the programming constructs covered in the course, a sheet listing common programming errors (and their solutions), and the course web site. They are not allowed to use any other materials, and the instructors will only answer questions about the problem (rather than helping students correct syntax errors, for example). In essence, these practica serve as programming exams that test the students' individual programming skill.

Students complete two lab practica over the course of the semester. In the first practicum, held in the middle of the semester, students use procedures, selection statements, and condition-controlled iteration. The second practicum, held approximately three weeks before the end of the semester, requires that students also use count-controlled iteration, arrays and file input and output. The problem statement from one version of the second practicum is provided in Figure 1. Since it is sometimes difficult to precisely describe required graphical output for a program, the practicum handout also contains an example of the required output.

3 Practicum Administration

Although the previous section describes the technical content of the lab practica, there are numerous administrative issues that need to be addressed when we use practica within a course. For clarity of presentation, we address the preparations required prior to each practicum,

Develop an Ada program that will do the following:

1. Display a *short* introductory message in the text window (do not require the user to press return!).
2. Open a graphics window that is 500 pixels wide and 300 pixels high.
3. Get ten left mouse button clicks from the user. After each mouse click your program should:
   a. Write in the text window the number of mouse clicks that have been recorded so far
   b. Store the x and y coordinates from each mouse click in an X array and a Y array
   c. After the first mouse click, draw a line (any color) from the previous mouse click to the current mouse click
4. Find the maximum value in the X array
5. Find the maximum value in the Y array
6. Create a file called "A:\Practicum_2a.dat" and write the information described in the format shown below. Note: do NOT write the string "Max X" in the file; write the **value** that is the maximum X value
7. Wait 5.0 seconds and close the graphics window.

Figure 1. Problem Statement from Practicum 2

the administration of the practicum itself, and post-practicum tasks.

3.1 Pre-Practicum

Before administering a practicum, we accomplish a number of administrative tasks. The most difficult of these tasks is developing between four and six versions of each practicum. Our motivation for using multiple practicum versions is twofold. First, we want students at adjacent workstations during a particular practicum to have different problems to solve. Although we are not concerned that our students will cheat, we do recognize the temptation presented by the ability of a student to glance at their neighbors' solution, especially in a high-stress environment. Second, we do not want students taking the practicum during a later session to have an unfair advantage over those students who take it earlier. Before administering a practicum to any students, we provide all of them with a list of the programming tasks from which their solution to that practicum will need to draw.

Of course, one of the largest concerns we have with using different practicum versions is ensuring that the different versions are of approximately equivalent levels of difficulty. Informally, instructors teaching the course review the different versions and raise any equity issues before the practicum is administered to the students. We provide a more formal statistical comparison in the following section.

For some practica, we also provide the students with a "checking tool" so they can evaluate their solution's performance. For example, for the problem statement provided in Figure 1 we provide the students with a tool that reads their output file and draws the corresponding lines, which gives the students an easy way to check their solution. We have found that these tools are particularly useful when the student programs are required to create file output, because students can then test their solutions graphically rather than examining a text file. These tools must obviously be developed before the practicum so that students can use them during the practicum.

Finally, because our students use blank floppies during the practicum (see the following sub-section), we need to ensure we have a sufficient number of floppies to distribute and that these floppies are blank. We use office administrative support rather than instructor time to meet this requirement, but it is clearly a consideration that needs to be addressed before the practicum is administered.

3.2 Practicum

Administering the practicum itself is straightforward, since the course instructors are only present to solve system problems and start and stop the practicum. We have implemented a number of rules for practicum administration that support the validity of our claim that the practica measure individual programming ability, however, and these rules merit some discussion.

For example, students are not allowed to reference any notes, books, or previous programs while they develop their practicum solutions. This rule clearly reflects the view that a practicum is a programming exam, though we have considered letting our students use previous programs they have developed as sources of help during the practica. These programs would certainly be a reasonable source of help, since many programmers use their earlier work as they develop new programs, but it leads to two critical problems. The first problem is ensuring that students only bring in their old programs rather than the programs of others or other materials. Verifying this by having each student bring one floppy and checking all the floppies manually (for example) would be an administrative nightmare. The second issue is even more important to us. Our main goal in the lab practica is to evaluate individual programming ability. Students develop their other programs in the course collaboratively, though, so letting them use programs they developed with others to help them as we try to evaluate their individual abilities seems inconsistent. For these reasons, we do not let the students bring any materials into the lab practica.

We do recognize, however, that requiring our students to solve the practicum problems without any references whatsoever might inhibit our ability to accurately measure their programming ability. Therefore, we provide each student with a booklet containing the syntax and an example of each Ada programming construct used in the course, as well as a sheet listing common Ada errors and their most common causes. In addition, students are allowed to access the course web site, which contains a program template builder, sample calls to numerous graphics procedures, and example code from the Ada textbook.

The students are not allowed to bring any of their own materials into the lab, and we also prohibit them from accessing their network drives or any e-mail systems during the practica. We are convinced that, given our academic environment, the students follow our instructions, but this rule could be strictly enforced by manually disconnecting the computers in our lab from the network. This could be accomplished fairly easily, with no impact to the students if the course web page is also hosted on each hard drive during the administration of each practicum.

Another rule that we have implemented to treat the practica as programming exams is that instructors are only allowed to provide their students with minimal help during the practica. Instructors can answer questions about the problem statements, but provide no other help as the students try to develop their solutions. Instructors can step in to solve system problems or to address compilation problems that reflect a bug in the compiler or in the system configuration, but they cannot provide any other help to the students. This is a difficult rule for the students, and even for some instructors, but we feel the resulting solutions more accurately reflect each student's individual skills.

Finally, we note that students are not allowed to save any of their work to the hard drive while they complete their practicum; they must save their work exclusively on the floppy disk we provide to them. This ensures that their solution can be easily collected at the end of the practicum, and also ensures there are no practicum solutions left on the hard drives when the later sessions of the practicum are administered. Requiring that all development be accomplished on a floppy could cause slower compilation, so we have configured the lab machines to place intermediate files on the hard drive even though the source code is located on the floppy. We also encourage our students to save regularly during the course of the practicum so they do not lose their work in case system problems occur. At the conclusion of the practicum, most instructors copy student solutions on a separate floppy as well, so instructors leave the lab with two copies of each student solution. This approach helps avoid the requirement to have a student retake the practicum if their floppy is no longer readable when the instructor is ready to grade the student solutions.

3.3 Post-Practicum

After the students have completed their practicum solutions (or time has expired), the instructors must of course grade the student solutions. Typically, instructors copy all the student solutions to a single folder on their hard drive, then execute batch files we have developed to compile, build, and print each student's solution. At that point, instructors simply need to run and grade each student's program just as they grade other programs in the course. To ease the grading load, particularly for graphical programs, we have developed some automated tools to help with the grading of both the practica and other programs the students generate during the course. Essentially, these tools run each program with a given set of inputs, generating textual outputs in a separate file. This file can then be checked for correct program performance, which can be quicker than manually running each student program and examining the graphical output.

4 Statistical Analysis

In the previous section, we mentioned that one of our largest concerns with the lab practica is ensuring that the different versions of each practicum are of comparable difficulty. We would like to assume that this is the case based on our informal instructor review of each practicum before it is administered. In this section, we use statistical analysis to check whether or not this is a valid assumption. Specifically, for each practicum, we compare the distributions of student performance on each version to determine whether any differences in the means of those distributions are statistically significant.

Our dataset includes scores for 509 students from the Spring 2000 semester (507 scores for the first practicum). We used four different versions (a, g, q, and s) of Practicum 1 and six different versions (a, f, k, n, s, and w) of Practicum 2.

First, we examine the summary statistics (see Figure 2.) for the different practicum versions. Most obviously, we note that the scores on the second practicum were lower than the scores on the first practicum. We believe this was caused by the more advanced constructs required on the second practicum, as well as the students' workloads in their other courses near the end of the semester. We also note that, for each practicum, the means appear to differ somewhat across the different versions, but the standard deviations are relatively large as well. We therefore need to defer making any judgements about differences between the means until after we have completed a more formal statistical analysis.

	Practicum 1				*Practicum 2*					
	a	g	q	s	a	f	k	n	s	w
Mean	63.65	60.96	62.04	60.89	53.75	55.57	54.18	57.23	53.69	54.89
Std Dev	12.81	13.17	12.64	12.92	15.66	15.46	17.60	13.90	15.37	16.87
Count	129	135	129	114	88	84	82	84	87	84

Figure 2. Practicum Summary Statistics

Before we begin the comparison of means for the different practicum versions, we note that the majority of statistical analysis techniques are based on the assumption that a particular population has been sampled. The statistical tests then help us quantify the strength of our hypotheses based on the characteristics of the sampling process and the resulting distributions. In our case, we have scores for the entire population of students completing each practicum, so in a sense we have not sampled at all. We choose, however, to treat each such distribution as a sample of the population of all students who could have taken that version of the practicum, either in the current semester or in the future. Based on this perspective, applying standard statistical tests to conduct our comparison is an appropriate approach.

A four-way Analysis of Variance (ANOVA) for Practicum 1 or a six-way ANOVA for Practicum 2 would be common methods for comparing the means of the various score distributions. One of the underlying assumptions of ANOVA, however, is that the distributions being compared were drawn from normal distributions [2]. To check this assumption, we check the normality of the distributions being compared; if those distributions are not normal, we are unwilling to assume they were drawn from normal distributions. To formally check the normality assumption, we conduct the Kolmogorov-Smirnov test for normality on each distribution [4]. The Kolmogorov-Smirnov test tries to reject a null hypothesis that a particular distribution is normal (thereby strongly implying that it is non-normal). When we apply the Kolmogorov-Smirnov test to the distributions, we are able to reject the null hypothesis for all of the Practicum 1 distributions, implying that each of the distributions is non-normal. We are not able to reject the null hypothesis for any of the Practicum 2 distributions, so we cannot infer non-normality of the distributions. Because we come close to rejecting the null hypothesis for several of these distributions, however, and because we already need to use a test other than ANOVA for the Practicum 1 data, we choose to be conservative and select a statistical test that does not assume normality. Although ANOVA tests are very robust against violations of the normality assumption, there are other, more suitable statistical tests for the distributions we want to compare.

One such test for comparing multiple distributions is the non-parametric Kruskal-Wallis test. This test does assume that the distributions are independent of each other (certainly a valid assumption since the distributions represent the performance of different sets of students), but makes no assumptions about the normality of the distributions. The test uses the null hypothesis that the samples (e.g., distributions) come from identical populations and the alternative hypothesis that they come from different populations [5]. If the test statistic exceeds its critical value for a particular statistical significance (we used the common 0.05 level), the null hypothesis can be rejected.

When we perform the Kruskal-Wallis test on the Practicum 1 data, the resulting p value is 0.208, far too high for us to reject the null hypothesis. We take this result as strong statistical evidence that the different versions of Practicum 1 are of equivalent difficulty. When we perform the test on the Practicum 2 data, our p value is 0.788, leading us to infer that the multiple versions for Practicum 2 are also of equivalent difficulty.

Although the above Kruskal-Wallis test results indicate that the different versions for each practicum are of comparable difficulty when examined as a group, we would also like to determine whether or not there are statistically significant differences between pairs of practicum versions. While statistically significant results from the Kruskal-Wallis test would more strongly indicate the need for this pairwise comparison, our approach is of some interest, particularly for those pursuing similar analyses. To investigate further, we conduct pairwise comparisons between the distributions. This results in six comparisons for the Practicum 1 data and 15 comparisons for the Practicum 2 data.

Unfortunately, there are well-known problems with conducting large numbers of pairwise comparisons within a set of distributions; the more such comparisons we conduct, the higher the probability that we will incorrectly reject the null hypothesis in one of the comparisons [2]. Fortunately, numerous statistical tests have been developed to avoid this problem, including the Scheffé test. This test allows large numbers of pairwise comparisons while removing the risks of spurious rejection of the null hypothesis. We used the Scheffé test to conduct pairwise comparisons for both the Practicum 1 and Practicum 2 data, and did not find any statistically significant differences between pairs. This is of course not surprising based on the Kruskal-Wallis results.

There is still further analysis required, however. Although the analysis results above imply that the practicum versions

are of comparable difficulty, it could be the case that one of the versions is harder than the others, but was by chance administered to a stronger group of students overall. To examine this possibility further, we take a closer look at the groups of students who took each version. Although the versions are in essence randomly distributed to students, it is still possible that the population of students taking a particular version could consist of stronger students in the course than the populations for the other versions. We can check this by examining the results of other evaluation techniques in the course. Since student performance on the practica is most strongly correlated with their performance on tests and the final examination [1], we selected those evaluation techniques for further study.

To perform this investigation, we partition the results of each of the evaluation techniques by practicum version (one partitioning for Practicum 1 and another for Practicum 2). We then perform either the Kruskal-Wallis test to check for differences across all versions, or the Scheffé test to check between specific pairs, on the resulting partitioned distributions. If we found that the population of students for a particular practicum version had a higher final examination mean, for example, we would have some support for the claim that the versions were not of equivalent difficulty in spite of the comparable means for those versions. We partitioned the evaluation methods mentioned above for each practicum and conducted both Kruskal-Wallis and Scheffé tests on the resulting distributions. None of these tests yielded statistically significant results, again implying that the different practicum versions were of comparable difficulty.

5 Conclusions

Instructors teaching courses with a significant programming component are faced with the difficult task of evaluating the individual programming ability of each student. In this paper, we described how we have incorporated lab practica into an introductory computer science course to help us accomplish this task. We provided details about our administration of these practica and discussed specific issues that need to be addressed when implementing such an evaluation method. We showed through a statistical comparison that the different versions of each practicum were of comparable difficulty, and we presented a sound comparison approach for conducting more in-depth analyses.

Our approach to the lab practica is continually evolving. In the Fall 2000 semester, for example, we will be administering 3 practica (two 45 minute practica and one 90 minute practicum). Our motivation for doing this is to lessen the impact of a student doing poorly on a single practicum while also reducing the stress associated with taking each practicum. In addition, we recognize that many programmers reuse their old code when developing new programs, an approach we specifically prohibit in our administration of the lab practica. While we believe this decision better supports our ability to evaluate each individual's programming ability, we are looking for an effective way to let students use their old code without diminishing our ability to perform the evaluation.

We will also be faced with a difficult practicum administration problem in the near future. Our institution is likely to move to laptops for all our students in the next few years, with the probable loss of the dedicated labs we currently use to administer our practica. In an environment in which each student carries their laptop into the practica, it is not clear how we can control the materials they use on the practica as we can now.

We have been pleased with the effectiveness of the lab practicum approach, to the point of incorporating lab practica into other courses. Our simulation course now includes a practicum using ProModel, and our information warfare course includes a "hackticum" where students attempt to identify security holes in a particular operating system configuration. Both of these courses use a timed, closed environment with administration rules similar to those described above. Lab practica seem to provide an effective method for evaluating each student's individual skills, and the approach is general enough to be applicable to a wide variety of courses.

6 Acknowledgments

Many thanks are due to Larry Merkle for his insightful comments, which strengthened this paper considerably.

References

[1] Chamillard, A.T. and Braun, K.A., Evaluating Programming Ability in an Introductory Computer Science Course, In *Proceedings of the Thirty-First SIGCSE Technical Symposium on Computer Science Education*, Austin, Texas, March 2000.

[2] Cohen, Paul R., *Empirical Methods for Artificial Intelligence*, The MIT Press, MA, 1995.

[3] Davis, B.G., *Tools for Teaching*, Jossey-Bass Publishers, San Francisco, CA, 1993.

[4] Neter, J., Wasserman, W., and Whitmore, G.A., *Applied Statistics*, Allyn and Bacon, Inc., Boston, MA, 1978.

[5] TexaSoft's Statistics Tutorial – Kruskal-Wallis Test, http://www.texasoft.com/winkkrus.html