Identifying Programmer Ability Using Peer Evaluation: An Exploratory Study

Jeffrey C. Carver  
University of Alabama  
Box 870290  
Tuscaloosa, AL 35487  
1-205-348-9829  
carver@cs.ua.edu

Lorin Hochstein  
USC Information Sciences Institute  
3811 N. Fairfax Drive, Suite 200  
Arlington, VA, USA 22203  
1-703-812-3710  
lorin@east.isi.edu

Jason Oslin  
University of Alabama  
Box 870290  
Tuscaloosa, AL 35487  
1-205-348-6363  
oslin002@crimson.ua.edu

ABSTRACT
Programmer ability is important to assess in both research and professional settings, but it is very difficult to measure directly. The objective of this study was to measure the ability of programmers to accurately rate the programming ability of their peers. The participants were computer science students in a senior-level undergraduate programming languages course. The peer rating of ability on a 5-point scale was compared with the average programming assignment grade. The Spearman correlation rho value was between .432 and .684 depending on which measure of central tendency was used (mode, median, mean, weighted mean), compared with .48 when using individual assignment data as a predictor. Initial results indicate that peer-ratings are a promising mechanism for identifying the programming ability of a set of developers. This study leaves many open questions that will be the subject of future studies.

Categories and Subject Descriptors
D.2.8 [Software Engineering]: Metrics - process metrics, D.2.9 [Software Engineering]: Management - productivity

General Terms
Management, Measurement, Experimentation, Human Factors

Keywords
Expertise, peer evaluation, programming ability

1. INTRODUCTION
A famous early study by Sackman and Grant suggested that programmer ability can vary at a ratio of 28:1 [16]. Although Prechelt argues that typical variation is closer to 4:1 [15], even this lower estimate is a large enough effect that, if we could determine how to transfer the skills of the high-performing developers to lower-performing ones, it would improve productivity by an order-of-magnitude. This improvement would qualify as one of Brooks' silver bullets [2].

This large variation in programmer ability is well-known to empirical software engineering researchers: it is one of the reasons why so few of our studies achieve the recommended power of 0.8 (i.e., 80% probability of obtaining a statistically significant result assuming there is a genuine difference among treatments) [7]. This large variation is also well-understood in the software engineering practitioner community. Our impression is that much more care goes into selection of programmers than, say, programming language or development process, because the quality of the programmers is believed to have a much larger impact on project outcomes.

A major challenge for both researchers and practitioners is that it is very difficult to measure programmer performance directly. This problem is well-known to empirical researchers, who are often forced to use simplistic measures such as years of experience or “professional vs. student” to control for ability. They know that these measures are inadequate, but no other measures present themselves. In industry, where the stakes of being able to recognize programming ability are much higher, some companies resort to elaborate puzzle-based interviews to try to identify potentially high performing software developers [14].

The problem is that we do not have a good way to operationalize programming ability. While it seems likely that a professional software developer will perform better than a first year computer science student, we do not have good evidence that a particular professional with ten years of experience will perform better than another professional with five years of experience. We also know that simple metrics are not adequate. We cannot measure, say, lines of code developed per week to identify the more productive programmers from the less productive ones.

The central hypothesis of this paper is that, even if programming ability cannot be operationalized, developers knows it when they sees it. If the reader considers some of the better developers they have worked with in the past, he or she is likely to be able to name them based on personal judgment and experience. This paper describes a study that tests the hypothesis that subjective judgment about the ability of our peers is a reasonable predictor of their ability.
2. RELATED WORK

There has been extensive study comparing how experts and novices perform software tasks (page limitations do not permit a full literature review, which appear in other papers [18], [19]). Such studies typically rely on programmer experience as a proxy for expertise. However, as Sonnenage et al., note, most studies use years of performance as a proxy for expertise, when the underlying construct of interest is actually high performance [18]. Host et al. tested this hypothesis by comparing software professionals with students. Their study, involving a software engineering judgment task, found no statistically significant difference between students and professionals [10].

Host et al.’s result suggests the importance of identifying factors other than student/professional for predicting performance. Carver et al. sought to identify behavior in programmers’ backgrounds that explain variation in performance on finding defects in inspections [4, 5]. Dehadi and Bornat claimed to have identified a test that can identify programming aptitude among novices without any programming experience [6], but a subsequent replication produced a negative result [13].

This paper focuses on peer evaluations and self evaluations. Bryant conducted a closely related study of expertise ratings in the context of pair programming. The participants each rated their own ability and were rated by their peers. The results of this study indicated that 79% of the high ability programmers underrated their own experience, while 50% of the low ability programmers overrated their own ability [3]. However, as in previous studies, Bryant used “length of tenure” (i.e. years of experience) as an objective metric for ability.

Psychologists have also studied the accuracy of assessments. In terms of the accuracy of self-assessments, Kruger and Dunning report that novices tend to overestimate their own ability because they lack the experience to recognize their own limitations. In addition, experts tend to overrate the ability of their peers because an expert assumes that their peers are as qualified as they are [11]. In terms of the accuracy of peer assessments, Holzbach found that peer ratings were better correlated to ratings of superiors than to self-ratings, suggesting that self-rating and peer-rating are actually measuring different things [9]. Finally, in a study of peer evaluations of police officers, peer rankings (i.e. ordering peers from 1..n) were very reliable. They were more accurate than peer ratings (i.e. scoring based on some predefined scale) [12]. Peer ratings seem to be a valid method of evaluation. Conversely, people may be better able to determine the relative ability of their peers rather than providing an absolute rating.

3. STUDY OVERVIEW

The main goal in this study was to determine how accurately students could judge the programming ability of their peers. Stated in GQM form [1], the goal of our study is to:

Characterize peer assessment of programmer ability from the point of view of a researcher in the context of an undergraduate computer science course

We had two requirements for conducting this study in the context of an undergraduate course. First, the course should be late enough in the curriculum that most students would be familiar with the programming ability of their classmates. Second, the course should have individual programming projects, graded by an objective (i.e. not involved in the study) instructor, upon which to compute an objective rating of ability.

To meet these requirements, we chose a senior-level programming languages course at the University of Alabama. Students developed five individual programming projects throughout the semester: an SSH log parser in Perl, two sets of exercises in F#, a program interpreter in Java, and an RSS reader in Java. The professor who taught the course is not an author of this paper and was not involved in the study: this was made clear to the students.

The remainder of this section details the study design. Section 3.1 introduces the hypotheses that were evaluated during this study. Section 3.2 provides a detailed description of the study design.

3.1 Goals and Hypotheses

The overall goal of this study was to determine whether or not students could accurately rate the programming ability of their fellow classmates. We also wanted to know how accurately they could rate their own ability relative to their classmates. Based on these goals and the related work, we posed two hypotheses that guided the study design and analysis:

H1: Developers are able to accurately predict the programming ability of their peers

H2: Low-performers tend to over-estimate their ability

3.2 Study Design

In order to evaluate these hypotheses, two important pieces of information were needed from each participant. First, it was necessary to obtain an evaluation of each participant’s ability as judged by themselves and their fellow classmates. Second, it was necessary to obtain some measure of the participants’ true programming ability. After these two measures were obtained, they could be compared with each other to evaluate the hypotheses.

To obtain the first important piece of information, classmate peer evaluations, at the beginning of the semester each participant was provided with a form to complete which contained a list of the names of their classmates. They were then asked to rate the ability of each classmate relative to the abilities of the other students in the course. The participants were told that they did not need to provide a rating for any classmate with whom they were not familiar. To facilitate the rating, the participants were instructed to use the following rating scale:

A = student is in the top 20% of the class
B = student is in the top 21% - 40% of the class
C = student is in the top 41% - 60% of the class
D = student is in the top 61% - 80% of the class
E = student is in the bottom 20% of the class

Along with each rating, the participants were instructed to provide an estimate of their confidence in that rating. The confidence estimate was rated on a 3-point scale (C-Low, B-Medium, and A-High). A copy of this form appears in Appendix A.

To obtain the second important piece of data, a measure of true programming ability, we had the students perform a set of programming tasks across a range of programming languages, and we used an expert judge to assess the quality of their work. To accomplish this, we used the participants’ grades on a series of
 programming assignments conducted throughout the semester. Each programming assignment evaluated the participant’s ability to implement various programming features in different programming languages (as described in Section 3). The professor of the course weighted each assignment appropriately (as required by the syllabus, not by this study) to obtain a final programming score for each participant. This final score was on a 0-100 scale. We then ranked each student’s true ability relative to their classmates using the same A-E scale they used to rate their peers. The transformation of number grades to the letter scale is shown graphically in Figure 1.

Figure 1 – Coding of assignment grades

In addition to these two main measures, at the end of the semester, the students were given a survey to help us obtain further insights and properly interpret the results. The survey focused on understanding whether the participants thought their performance in the course was indicative of their true ability, whether they thought the rating scale was valid, whether they thought they and their classmates were good judges of programming ability, and any other information that may be important. Because this was an exploratory study aimed at identifying interesting questions for future research, we use an alpha of .1 for the hypothesis tests.

4. RESULTS

Section 4.1 provides an overview of the data collected during the study along with some descriptive statistics. Then, Section 4.2 discusses the results related to H1. Section 4.3 describes the results related to H2. Section 4.4 discusses some observations about the participants in the study. Section 4.5 describes observations from the post-study questionnaire. Section 4.6 provides some discussion of the results. Finally, Section 4.7 explains the threats to validity that were present in this study.

4.1 Description of the data

First, we will describe the data collection process and the level of participation of the potential participants. Of the 36 undergraduate students enrolled in the course, 18 participated in the peer-rating exercise at the beginning of the semester (the other students chose not to participate). By the end of the semester, 4 of the 36 students had dropped the course, so we have programming project grades from 32 students.

To provide context for the detailed results in the following sections, some descriptive statistics are in order. The first observation is that the students were not as familiar, at least by name, with their classmates as we initially assumed. This fact is evidenced by the relatively small number of classmates rated by each participant. Figure 2 shows these relationships graphically. Each row represents a rater. Each column represents a ratee. A shaded cell indicates that the rater represented by the row provided a rating for the ratee represented by the column (dark borders indicate self-ratings). Another important observation about the peer ratings is the fact that they were skewed heavily towards positive ratings. As Figure 3 indicates, the vast majority of the ratings across all participants were either A (5) or B (4).

Figure 2 – Coverage of Peer Ratings

Figure 3 - Distribution of Peer Ratings

4.2 Hypothesis 1

Hypothesis 1 stated that developers are able to accurately predict the programming ability of their peers. To evaluate this hypothesis, we used two variables for each participant: a grade and a peer rating.

The grade variable was used as described in Section 3.2. The peer variable was computed by first converting the letter scores to numerical scores (A=5, B=4, C=3, D=2 and E=1). We also converted the grade to a numerical score. For each participant,
the ratings provided by the other participants were aggregated. (Note that self-evaluation was excluded from the aggregation as we were interested only in peer-ratings). As this was an exploratory study and we were interested in understanding which measures are most appropriate, four different methods of aggregation were computed: mean, weighted average, median and mode. Mean, median and mode were calculated in the standard fashion. The weighted average was calculated by taking into account the confidence estimates provided by each participant along with their rating. The formula for computing the weighted average was to multiply each rating by the confidence and then divide but the sum of all of the confidence values. This formula ensures that the peer rating value remained in the given 5-point scale, but provided extra weight to those ratings that were given with more confidence.

To evaluate the hypothesis, we conducted a Spearman Rho correlation of the grade with each of the four measures of peer ratings. The results in Table 1 indicate that mode was the most accurate predictor of the actual grade. Figure 4 shows a scatter plot of the mode data with a linear regression trend line. Ideally, this line would have a y-intercept of x=0 and a slope of 1. The results of the linear regression are shown in Table 2.

Table 1 - Comparison of Ratings to Actual Scores

<table>
<thead>
<tr>
<th>Central Tendency</th>
<th>Spearman</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>.432</td>
<td>.039</td>
</tr>
<tr>
<td>Weighted Average</td>
<td>.432</td>
<td>.04</td>
</tr>
<tr>
<td>Median</td>
<td>.489</td>
<td>.018</td>
</tr>
<tr>
<td>Mode</td>
<td>.684</td>
<td>.002</td>
</tr>
</tbody>
</table>

Table 2 - Linear model of peer ratings

<table>
<thead>
<tr>
<th>Estimate</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slope</td>
<td>1.23</td>
</tr>
<tr>
<td>Intercept</td>
<td>-2.34</td>
</tr>
<tr>
<td>Adjusted R-squared</td>
<td>.3997</td>
</tr>
</tbody>
</table>

The slope is 1.23, which is not far off from 1.0, and the result is statistically significant. The adjusted R-squared, at .3997, shows a decent effect size: peer ratings account for almost 40% of the total observed variance in the data set, as seen in scatterplot by the distance the points are from the trend line.

This result suggests that peer ratings do have some predictive power about programmer performance. But it is difficult to gauge the quality of peer rating in an absolute sense. For a baseline, we can use more detailed grading information from the course. For each student, we have the grades from five separate programming assignments. In our analysis above, we used an aggregated measure of this data. However, we can also use this measure as a basis for comparison. We can compare how closely peer ratings match assignment grades with how well assignment grades match each other. We calculated the mean Pearson correlation across all pairs of the five assignments, yielding Pearson correlation of 0.476. This result suggests that peer evaluation is better than previous assignment grades in the same course as a predictor of performance when using the mode, and is about the same when using the other measures of central tendency.

A second analysis related to this hypothesis was to determine whether participants were more accurate in rating high-quality peers. To conduct this analysis, we computed the difference between the peer rating and the actual score (based on the programming assignment grades). Because mode was the most accurate predictor of ability, we used it as the value for the peer rating. As Figure 5 indicates, the ratings for the best students, those in the top 20% of the class, was better than for the other students. Of course, it is possible that this result is due to the fact that the ratings were skewed towards the high end overall. Nevertheless, this observation is interesting and should be studied further in replications of this study.

In examining the data distributions shown in Figure 1 and Figure 3, we observed that both the grades and the ratings were highly skewed towards the upper end of the range. It is possible that students had trouble rating their peers into quintiles as we asked and rather defaulted to rating them based on grades. Based on this observation we reanalyzed the data making two different assumptions to test alternate hypotheses.

First, instead of a student’s actual grade being based on what quintile of participants he or she fell into, we recoded the actual grade to be based on what quintile of the grade range their overall programming grade fell into, i.e. grades between 80 and 100 were scored into the A group, 60-80 in the B group, and so on. We
conducted the same correlation analysis on this new distribution of grades to see if there was any difference. The results showed that for all 4 central tendency measures of peer-rating, the correlations were weaker (Table 3).

Table 3 - Comparison of Ratings to Modified Actual Scores (Quintile based on grading scale)

<table>
<thead>
<tr>
<th>Central Tendency</th>
<th>Spearman</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>.643</td>
<td>&lt; .001</td>
</tr>
<tr>
<td>Weighted Average</td>
<td>.683</td>
<td>&lt; .001</td>
</tr>
<tr>
<td>Median</td>
<td>.678</td>
<td>&lt; .001</td>
</tr>
<tr>
<td>Mode</td>
<td>.698</td>
<td>&lt; .001</td>
</tr>
</tbody>
</table>

Second, instead of the actual grade being based on a quintile of either the grade or the ranking within the class, we recoded the actual grade to match the traditional definition of the corresponding letter grades, i.e. 90-100 = A, 80-90 = B, and so forth. Recomputing the correlations with this new definition of actual grade led to slight improvements in the correlations over the initial analysis. The results are shown in Table 4.

Table 4 - Comparison of Ratings to Modified Actual Scores (Based on traditional grading scales)

<table>
<thead>
<tr>
<th>Central Tendency</th>
<th>Spearman</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>.643</td>
<td>&lt; .001</td>
</tr>
<tr>
<td>Weighted Average</td>
<td>.683</td>
<td>&lt; .001</td>
</tr>
<tr>
<td>Median</td>
<td>.678</td>
<td>&lt; .001</td>
</tr>
<tr>
<td>Mode</td>
<td>.698</td>
<td>&lt; .001</td>
</tr>
</tbody>
</table>

The preceding analysis does not consider self-ratings. Out of the 16 raters, only 8 provided a self-rating. These ratings are shown in Table 5. If we compute the Spearman Rho correlation, we obtain $r = -0.655$ ($p = .078$), a negative correlation! This result is examined in more detail in the next section.

Table 5 - Self Ratings

<table>
<thead>
<tr>
<th>ID</th>
<th>Grade</th>
<th>Self-rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>D</td>
<td>A</td>
</tr>
<tr>
<td>13</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>16</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>19</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>20</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>22</td>
<td>A</td>
<td>E</td>
</tr>
<tr>
<td>26</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>27</td>
<td>B</td>
<td>A</td>
</tr>
</tbody>
</table>

4.3 Hypothesis 2
Hypothesis two stated that lower performers tend to over-estimate their ability. Of the eight participants that provided self-ratings, there are only two with grades of C or less (11 and 26). Both of these rated themselves A, but two samples are not sufficient to draw conclusions.

Therefore, we can test an alternate hypothesis:

$H2.1$ There is an inverse relationship between programmer ability and programmer’s perception of ability.

To test this hypothesis, we compute a linear regression and check the slope of the line of best fit.

As in the previous section, we first convert the letter scale to numbers (A=5, B=4, C=3, D=2, E=1) and then fit a linear model with self-rating as the independent variable and grade as the dependent variable. The statistical analysis is shown in Table 6, and the scatter-plot with regression line is shown in Figure 6.

Table 6 - Linear Model of Self Ratings

<table>
<thead>
<tr>
<th>Estimate</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slope</td>
<td>-0.41</td>
</tr>
<tr>
<td>Intercept</td>
<td>5.6</td>
</tr>
<tr>
<td>Adjusted R-squared</td>
<td>.078</td>
</tr>
</tbody>
</table>

Figure 6 - Grades vs. Self-Rating
As can be seen from the results, the trend line is indeed downwards, but there is insufficient power to obtain statistically significant result. One hypothesis from a visual inspection of is plot is that while low performers tend rate themselves highly, there is more variation in the self-rating of high-performers.

4.4 Variability across participants
In addition to mean ability of peers to identify programming ability, we also examined the variability in these evaluations.

In Section 4.1, we used the mode peer rating to determine how well the students did in predicting the programming ability of their peers. We can also look on a per-participant basis to see how well individuals did. We examined the median Pearson correlation of individuals, consider only individuals that rated three or more peers. Out of the 18 participants, 11 provided ratings for three or more peers. Figure 8 shows a wide variation in agreements between peer ratings and assignment grades, with a median Pearson correlation of 0.50. The boxplot suggests that the mean correlation is greater than 0. A one-sample t-test (which tests the hypothesis that the mean value is greater than zero) yields a $p=.0077$, although here we are probably pushing the statistical machinery farther than it was designed (by running a t-test on medians of Pearson correlations).
To explore possible explanations for this variation, we looked at two variables: rater grade, and rater confidence. Table 8 shows an analysis of variance which suggests a statistically significant effect due to rater grade, and a significant grade-confidence interaction, but no effect due to rater confidence. Figure 7 shows an interaction diagram with the mean peer rating error (|assignment grade – peer rating|) for all subgroup combinations of confidence and rater grade. We hypothesized that students with better grades would tend to be better raters overall, and raters with higher confidence would tend to be more accurate. Therefore, we expected to see a diagram with three downward-sloping lines (error decreasing as rater grade increased), with the higher confidence line at the bottom of the graph and the lower confidence line at the top of the graph (lower error with greater confidence). In the actual plot, we see that the effect of confidence interacts with the effect of the rater’s grade. In particular, high confidence ratings from raters with high grades were the most accurate, where high confidence ratings from raters with low grades were the least accurate.

Table 8 - ANOVA of error vs. rater grade, confidence

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>SS</th>
<th>MS</th>
<th>F</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rater grade</td>
<td>1</td>
<td>0.25</td>
<td>7.516</td>
<td>10.376</td>
<td>.0020</td>
</tr>
<tr>
<td>Confidence</td>
<td>1</td>
<td>0.0085</td>
<td>0.001</td>
<td>0.0011</td>
<td>.97</td>
</tr>
<tr>
<td>Rater grade x Confidence</td>
<td>1</td>
<td>0.25</td>
<td>7.023</td>
<td>9.697</td>
<td>.0027</td>
</tr>
</tbody>
</table>

4.5 Post-test questionnaire

At the end of the study, we administered a post-test questionnaire, which included the following questions:

1. Did you think your performance on these assignments accurately reflected your programming ability?
2. Do you think a five-letter scale (A-E) is a reasonable way to rate someone’s programming ability?
3. Do you think you are a good judge of your classmates’ programming ability?
4. Do you think you are a good judge of your own programming ability?

Fourteen of the participants filled out the post-test questionnaire; the results are shown in Table 7. While the sample is too small for statistical analysis, we made the following observations:

The majority of questionnaire respondents thought that assignment grades were a reasonable proxy for programming ability (Q1). The two most common complaints about the merits of judging programmer ability using programming assignments were time constraints and extenuating circumstances (illness, family concerns).

Most questionnaire respondents thought that the rating scale was suitable (Q2). Some acknowledged that there are many ways to judge ability and many factors to consider when doing so.

All questionnaire respondents except one thought that they were good judges of their own programming ability (Q4) and better judges of their own performance than their peers (Q3,Q5), an assumption not supported by the data. They tended to be less sure about the accuracy of their own ratings of peers (Q3), but seemed least confident in the ratings they received from their peers (Q5).
Many students cited unfamiliarity with one another as a reason that other people in the class would probably not rate them properly (Q5).

4.6 Discussion
As this was an exploratory study, we cannot draw firm conclusions from our results, but the data raises several interesting issues.

We were surprised by the low number of rating scores per participant. In a class of 36 students, the highest number of peer ratings (including self-rating) was 11, and the mean was 5.3. Speaking to some students afterward, one issue we discovered is that the students did not rank some classmates because they did not know them by their full name. We assumed that students would know each other’s name, so our rating form we used the name that appeared on the class roster. Some students knew classmates by nicknames or only knew their first name. This observation suggests that in future studies, it may help to have students’ pictures next to their names on the ratings form.

If the participants were doing complete ratings of their peers, we would have expected the distribution of counts to be uniform, since the peer ratings are based on quintiles. However, as shown in Figure 3, the ratings were heavily skewed towards higher performers, implying that the participants preferred to rate peers they thought were higher performers.

All the students who disagreed that programming assignments were a good judge of programming ability rated a 3 or below (bottom 60% of class). This may be explained by hypothesis 2 (low performers overestimating their ability) and cognitive dissonance: since the results of the study contradict a belief, the apparent contradiction is rationalized away by concluding that the performance evaluation is invalid. A rival hypothesis is that some otherwise talented programmers happened to do badly on this particular set of programming assignments. In a follow-up study, we plan to ask this question on both the pre-test and post-test, and look for a relationship between change in pre-test/post-test scores and performance on the programming assignments.

4.7 Threats to Validity
Construct validity. The most significant threat to validity in this study is construct validity [17], which takes many forms here. Because we do not have a good a priori measure of programming ability, we do not know how well ability is actually captured by an expert evaluation of quality in the form of high grades on the assignments. There is also a possible ceiling effect, as is suggested by the skewed distribution of the grades seen in Figure 1. Furthermore, we believe that software development ability involves a multitude of technical skills and social skills. Just as Gardner proposed the idea of multiple intelligences [8], we suspect that there are multiple areas of programming ability, which cannot be captured by observing a set of small programming tasks and taking a single measurement.

Internal validity. Since this study did not involve comparison across multiple treatments, many of the internal validity threats do not apply here. The main threat to internal validity is the low number of peer ratings provided by each participant.

External validity. We also have external validity threats typical of software engineering experiments conducted in the classroom. We have no real sense of whether peer evaluation would be equally accurate in professional software development settings, where the nature of interaction among programmers is very different than in an undergraduate computer science environment.

5. CONCLUSIONS AND FUTURE WORK
The results of this exploratory study suggest that peer ratings have some promise in being able to identify programmer ability. We saw that peer ratings are roughly as good at predicting programming assignment grades as using previous grade information. When we used the mode as our measure of central tendency, we saw a significant improvement over using grades. Certainly, further study is warranted.

One particularly interesting result was that the participants appeared to be much better at rating peers than rating themselves. This result contrasts strongly with the perceptions of the participants as measured in the post-test questionnaire: participants believed they were better judges of their own ability than other people were of them. The post-test questionnaire also suggested a “Lake Wobegon effect” where more students believed that they were better judges of others than others were of them.

Another interesting effect we observed in this paper is the bias of raters towards higher-performing individuals. We saw that participants only chose to rate a handful of their peers in the class, but the ones they rated tended more heavily toward ones they thought were better performers. In addition, the participants did better when rating higher performers. One theory is that people hold the correct belief that they are better judges of high performers than low performers, and feel more comfortable doing assessment of higher performers.

In this study, there were no consequences to being rated higher or lower in ability by a peer. However, once these ratings have consequences (e.g., peer evaluation used to determine pay raises), then this operationalize measure will undoubtedly fail, as there will be powerful incentives to try to game the system. This phenomenon, known as Goodhart’s Law was stated by Strathern as: “When a measure becomes a target, it ceases to be a good measure” [20]. Therefore, we do not see this measure as being useful in any context where the programmer has an incentive to receive a particular score. On the other hand, in a research context, where there are no incentives for being assigned a particular score (e.g., for use in blocking in controlled experiments, or for helping to identify high-performers for future research on programming ability), then we think this may be a useful tool.

We see this pilot study as an initial stepping stone into further research on programming ability. We envision three stages:

1. Characterize high-performers: develop methods for identifying different levels of expertise among software developers.
2. Understand programming ability: Study how higher performers are different from lower performers, such as identifying factors in their background or identifying development strategies that they employ.
3. Improve competent programmers: Determine how to transfer the skills of the highest performers to those who
do not perform as well but are receptive to improvement.

The next step is to replicate this study in another experiment in a classroom environment. If the results hold, a further step would be to try this study in an industrial environment. As our current peer rating scale and reference measurements are based on classroom settings, a transition to industry will require modification of the measurement instruments.

Based on the lessons learned during this study, we will make the following changes in our replication:

- Re-designing the rating form to better facilitate student peer-rating. This re-design may include adding photographs of the students and/or including nicknames.
- Encouraging participants to provide ratings of all peers and not just those they know well. We will try to combat the observation in this study that students mostly rated peers that they had positive impressions of.
- Revisiting the definition of the A-E rating scale to determine whether its construct validity can be improved. For example, it is possible that students had a difficult time classifying their peers into quintiles. Asking them to rate their peers based on what grade they would receive may provide more accurate results.
- Investigating interacting variables that may affect ratings, such as social relationships/networks, gender and self-efficacy.

6. ACKNOWLEDGMENTS

We would like to acknowledge Dr. Nicholas Kraft for allowing us to use his course as a vehicle for conducting this study. We would also like to thank Jamie DeCoster of the Statistical Consulting service at the University of Alabama for help in designing the study and the analysis.

7. REFERENCES