Can University Marks Measure Programming Skills for Novice Programmers? An Exploratory Study

Fanghui Zha[1], Yong Wang[1,2], Liqiang Mao[3], Jin Liu[4], Xue Wang[1]*

[1] *School of Computer and Information, Anhui Polytechnic University, China*
[2] *Institute of artificial intelligence, Hefei Comprehensive National Science Center (Anhui Artificial Intelligence Laboratory), China*
[3] *Wuhu Education Examination Center, China*
[4] *International college, Chongqing University of Posts and Telecommunications, China*
fanghuizha@qq.com, yongwang@ahpu.edu.cn, 617814@qq.com, 1402793952@qq.com, 2270946013@qq.com

Abstract

Programming skills are the key ability of programmers. For novice programmers who are new and have little experience in programming, it is important to measure and improve their programming skills. In practice, novice programmers learn programming knowledge and coding ability at university, and course learning is the main way for novice programmers to improve programming skills. However, can university marks be used to measure the programming skills of programming novices? To answer this question, in a controlled experiment, we compare university marks of novice programmers with their performance in programming tasks. The results showed that there were 7 courses with significant correlation among the 13 courses, among which the software engineering course had the most significant correlation.

Keywords: Novice programmer, Programming skills, Course scores, Correlation analysis

1 Introduction

As Internet companies increasingly demand higher software quality, they also have higher requirements for programmer' programming skills. The complexity of conceptual structure in specification, design, and testing cannot be eliminated in the short term with better programming languages and better tools [1]. Although the software industry has flourished in the past 40 years, practitioners still lack effective methods to eliminate complexity, which has also led to the software industry still relying heavily on the programming skills of programmers in the process of industrialization [2-7].

However, for novice programmers who are studying at university and are about to enter society, course learning is the main way to improve their programming skills [8-9]. Therefore, it is more important whether these courses can measure the programming skills of novice programmers

Around 2013, our country introduced the concept of outcome-based education (OBE) to guide the reform of engineering education, making the concept of OBE practical in China. In today's higher education teaching, people pay more attention to the actual needs of the return of educational investment and actual output, and the OBE of outcome-based education has become the mainstream concept of educational reform in our country. The OBE concept pays more attention to the cultivation of students' corresponding ability, takes students as the center, conducts individualized evaluation, and evaluates the final results of students according to the requirements of different abilities in the course objectives, so that students can accurately understand and solve problems.

Therefore, under the OBE concept, the course score is obtained by the student after learning the knowledge related to the course and acquiring the corresponding ability. It represents the synthesis of the students' learning in the subject. Courses related to computer majors assess computer-related knowledge and abilities, which may involve students' programming skills [10-11].

Which courses of students are related to their programming skills and can measure programming skills? Regarding the exploration of measuring programming skills, some scholars have found many influencing factors, but it is not clear what the main influencing factors are and the specific influence value of each factor [12-14]. This paper mainly studies which courses can measure the programming skills of novice programmers, analyzes the correlation between different course scores and programming skills, and finds the courses with significant correlation. The contributions of this paper are as follows [15]:

- A test containing questions that measure programming skills.
- Preliminary evaluation of questionnaires from 124 undergraduate students.
- The seven most relevant courses and analyzed them.

2 Preparation Work

2.1 Definition of Programming Skills

A person's programming skills can be roughly divided into the general design ability of program, the detailed design

*Corresponding Author: Yong Wang; E-mail: yongwang@ahpu.edu.cn
DOI: 10.53106/160792642023112406002

ability, and the code specification ability. Among these, general design ability is the ability to divide the program into modules. The ability of detailed design is the ability to design the algorithm of each module in detail. The ability of code specification can be divided into the following five points: first, divide modules reasonably. Reasonable module division in the code can make later software maintenance and software reuse easier to achieve. Secondly, define the function clearly. There is a clear definition for each function in the coding process, which can improve the understandability of the program. Thirdly, code segmentation is clear and short [16-17]. In the coding process, you should be able to write clean code and present it in the form of paragraphs to improve code readability. Fourthly, the naming is accurate. Accurately define different functions and variables. Fifthly, the annotation is clear. Some important lines of the code are simply annotated to improve the quality of the code [15, 18].

2.2 Measure of Programming Skills

With the upgrading of the software industry, the demand for talents in the IT industry is increasing, and the number of novice programmers is also increasing. Therefore, how to fully understand the programming skills of novice programmers is very important. Obviously, course study is an effective means to improve the programming skills of students-novice programmers [19]. Therefore, it is necessary to conduct research on the relationship between their course scores and programming skills [20-21].

In the related research, there are many ways to measure programming skills [22]. First, it can be measured through code or program written. The better the programming skills, the faster and more accurate the programming. Then, it can also be measured by means of a survey questionnaire. Design a questionnaire about programming skills, such as how many projects you participated in or the total number of lines of code written, and then evaluate programming skills by answering these questions. The other is to organize the students through experiments, then give some programming questions, and test the students within the specified time to measure their programming skills [11].

This paper proposes to measure the programming skills of students through the analysis of their course scores. Thus, which courses are indicative of programming skills is what we need to study.

2.3 Impact of OBE on Measuring Programming Skills

The OBE is an educational concept based on learning outcomes or result-oriented, which clearly focuses and organizes each link in education, so that students can achieve the expected results in the learning process. The OBE education model focuses on analyzing the output of students' learning and reverse-designing the students' educational structure and related evaluation systems. Under the educational system of this concept, an effective teaching system can be better designed for the programming skills needs of novice programmers. Thus, they can better achieve the output effect of measuring programming skills.

For the experimental results, it can also confirm the above point of view. Course scores obtained in courses that implement the outcome-based education model in the teaching process can better reflect the level of students' programming skills. For example, the software engineering course shows a high correlation after analyzing the test results of novice programmers. This is also because the teacher adopts the OBE teaching mode in the teaching process of the software engineering course, which can make the students oriented by the learning results and pay attention to the cultivation of students' programming skills, so that their programming skills level can be easily measured.

2.4 Challenges of OBE

There are still some challenges in the OBE concept. These challenges are mainly summarized in the following five aspects: first, it is unfair to implement standardized evaluation for students of different backgrounds and schools of different qualities. Secondly, the preset ability level may not be applicable to different students. Thirdly, there is a lack of sufficient empirical research to show that OBE is effective. Fourthly, the implementation of OBE will increase the burden on teachers and universities. Fifthly, in the implementation process, it is easy to bundle other teaching reform programs together.

3 Collection of Curriculum Data Related to Programming Skills

3.1 The Process of Generating Data

According to the requirements of the outcome-based education concept, it focuses on the cultivation of students' abilities and a reasonable evaluation of the achievement of course objectives. Therefore, in the process of cultivating students of this major, it is necessary to formulate a complete evaluation system of course objectives. The evaluation of the achievement of the course objectives of this major is mainly based on the assessment method, and the final evaluation of the achievement of the course objectives is formed with reference to the student questionnaires and symposiums after the course. The detailed evaluation process is shown in Figure 1.

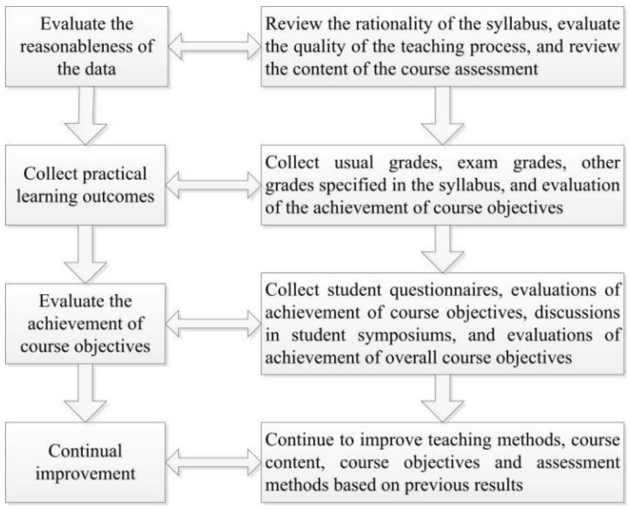

Evaluate the reasonableness of the data
Review the rationality of the syllabus, evaluate the quality of the teaching process, and review the content of the course assessment
Collect practical learning outcomes
Collect usual grades, exam grades, other grades specified in the syllabus, and evaluation of the achievement of course objectives
Evaluate the achievement of course objectives
Collect student questionnaires, evaluations of achievement of course objectives, discussions in student symposiums, and evaluations of achievement of overall course objectives
Continual improvement
Continue to improve teaching methods, course content, course objectives and assessment methods based on previous results

Figure 1. Flow chart of course goal achievement evaluation

The degree of achievement of the overall objectives of the course is calculated as follows:

C: the calculated value of the overall goal achievement degree of the course;

Wi: the achievement degree weight coefficient of course objective i;

OBi: calculated value of achievement degree of course objective i, OBi = score Bi of objective i / weight Wi of objective i.

Bi: the score of course objective i, Bi is obtained by the weighted summation of the achievement value Tj of each assessment sub-item of target i, and the weight of each assessment sub-item is set as wj, then:

$$B_i = \sum T_j * w_j . \tag{1}$$

Tj: the achieved value of the assessment sub-item, Tj = the average score of the assessment sub-item bj / the full score of the sub-item Mj.

Wj: The weight of the assessment sub-items, Wj is obtained by decomposing the weight of the course assessment methods stipulated in the syllabus, as detailed in the syllabus of each course.

Bj: The average score of the assessment sub-items. There are different types of bj scores. One is the assessment methods with specific scores such as assignments, tests, and examinations, and the other is the score obtained from the evaluation of the observation points, such as experiments, design reports, practice reports, and acceptance. (If it is a grade system, it needs to be converted into a percentage system for calculation).

According to the above relationship, the simplified calculation formula of C is as follows:

$$C = \sum OB_i * W_i = \sum \frac{B_i}{W_i} * W_i = \sum B_i . \tag{2}$$

3.2 Reasonableness Assurance of Data

The data sources of curriculum goal achievement evaluation include: syllabus, test paper, test paper scoring standard, test paper analysis table, teaching process quality evaluation table, etc.

First of all, after the completion of each course, the teacher should analyze and explain the data content, source and the collection method used in the evaluation of the course objectives, as well as the correlation among these data and the achievement of students' ability. Then it corresponds to these four reasonable guarantees:

Ensure that the syllabus is reasonable. It is necessary to ensure the rationality of curriculum objectives, the relevance of curriculum objectives and teaching content, and the rationality of assessment methods.

(1) Ensure that the teaching content is in line with the curriculum syllabus. In the teaching process, it is necessary to ensure that teachers are in accordance with the expected progress and teaching content to complete the predetermined objectives of the course.

(2) Ensure that the content of the assessment is reasonable to achieve the objectives of the assessment course. Teachers should analyze and explain how the assessment content connects to the course objectives to ensure rationality.

(3) Ensure the rationality of the evaluation of course objectives based on the questionnaire. It mainly analyzes the validity of the questionnaire, explains the content of the questionnaire, and deals with abnormal data.

4 Data Preprocessing

Due to the variety of students' courses, the dataset of students' course scores is relatively large, and there may be data missing, data noise, data inconsistency, data redundancy, and data repetition. Therefore, it is necessary to perform a certain degree of preprocessing on the data of course scores to make the analysis results of the whole data more authentic and reliable.

4.1 Data Filtering

We first generate the datasets of all course scores of all students in the entire major in the system, and then filter out the required course scores of juniors and seniors.

In the selected course scores of the third-year students, they are screened again to select the courses that they participate in collectively, including Java Programming, Software Engineering, Operating System, Big Data Storage and Processing, Computing Methods, Computer Composition and Structure, Database Principles and Applications I, Digital Image Processing II, Data Structure, Advanced Language Programming (C language), Scientific Computing Language (Python language), College English (1), Humanities and Social Sciences, a total of 13 courses.

In the course scores of the selected senior courses, their Data Structure and Advanced Language Programming (C language) are screened out as reference lines, and then Software Engineering, Scientific Computing Language (Python language), Database Principles and Applications I, Computer Composition and Structure, and Operating System are screened out, a total of 7 courses.

4.2 Data Cleansing

4.2.1 Missing Values Processing

Due to the process of acquiring information and data in the real world, there will be various reasons for data loss and vacancies. The treatment method for these missing values is mainly based on the distribution characteristics of variables and the importance of variables. In this experiment, some students (mainly students who have changed majors) have missing scores in some courses. After evaluation, the missing variables are of high importance to the whole experiment, so the method of filling statistics is used, and the average value is initially used filling. However, some of the repeating students' scores are also missing. Due to the high missing rate of such data and insufficient reference, such variables are deleted.

4.2.2 Inconsistent Data Processing

In the score table of a course, there are inconsistencies and contradictions in the scores of the same student. It shows that the student completes the course in different semesters; this is because the student failed to pass the course after taking the course for the first time, so he needs to retake the course. For this kind of data, we consider that students have multiple score values in the course because of retakes, and the retake courses are less rigorous than the formal courses, so the students' first course scores are taken as their final grades for the course to make the results more realistic.

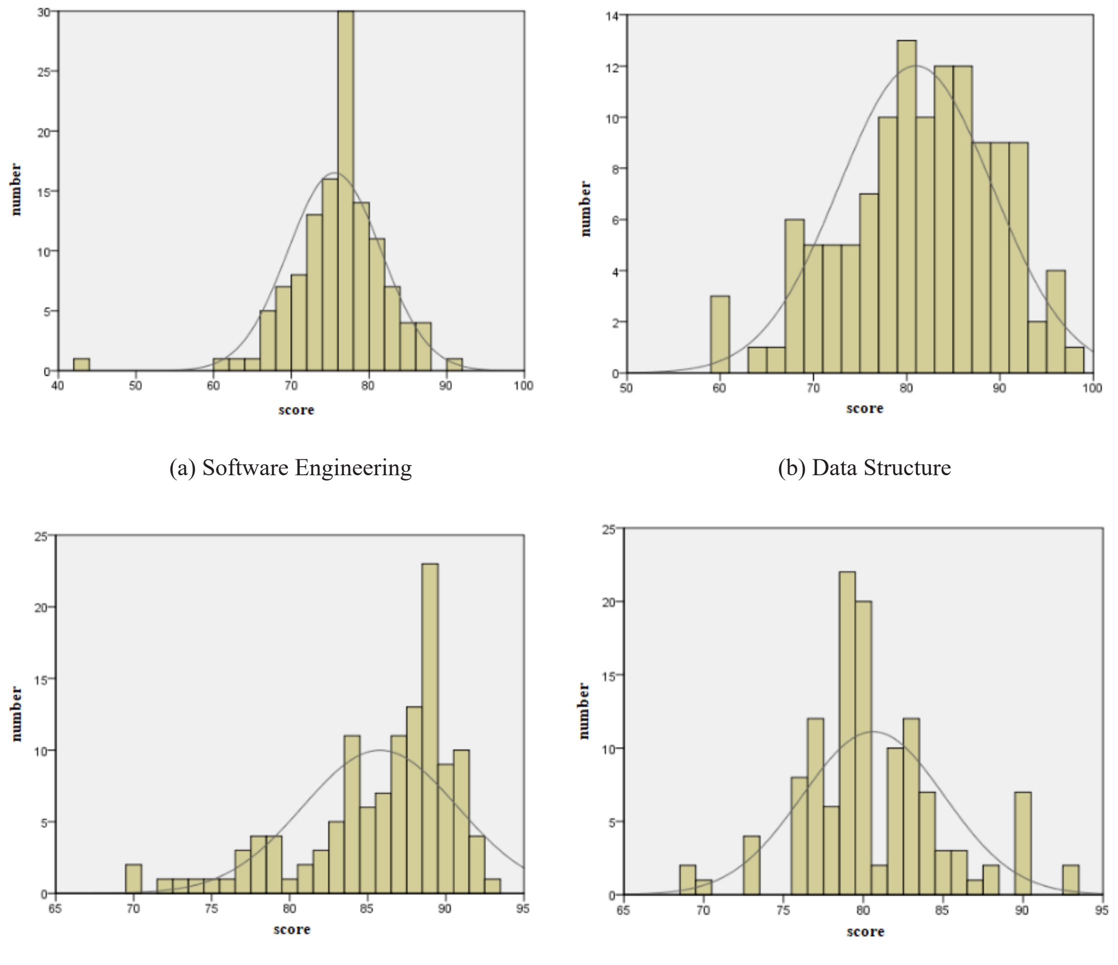
4.2.3 Data Redundancy Processing

Data redundancy is the situation where the amount of data or the number of attributes exceeds what is required for data analysis. In the exported data set of student course scores, there are other attributes such as the academic year, course program number, the course code, instructor, etc. These are redundant attributes that exceed the needs of data analysis, so they need to be deleted.

4.3 Data Description

After preprocessing the students' course scores, an operation can be performed on the data to visualize their distribution.

From the under three-year students' course score distribution Figure 2, we can see that the scores of these courses basically conform to the normal distribution, and the distribution is relatively reasonable. Similarly, the distribution results displayed after an operation of visualizing the course scores of senior students are also more reasonable.



(a) Software Engineering

(b) Data Structure

(c) Java Programming

(d) Computational Methods

Figure 2. Course score distribution diagram

5 Mechanism for Getting Coding Test Scores

In order to obtain reasonable student coding test results, the better the students' programming skills, the higher the correct rate of the tasks they solve. Therefore, when asking students to solve programming problems, each problem is set as a corresponding level, and only the answers of the test set that solve the question can be cleared and the score of this question can be obtained. The better the students' programming skills, the faster they can analyze and solve problems. Therefore, after students solve coding problems, we include efficiency points when grading students' coding problems. The faster a student successfully compiles and runs through the test set, the higher the efficiency score.

5.1 Basic Score

The basic score of a level is the set clearance score of each level. In order to better calculate and process students' scores, we set the basic level score of each level to 15 points, and there are 5 levels in total. Students can only get the basic score of the level after running the code program written in the platform and passing the evaluation.

5.2 Efficiency Score

Efficiency score refers to the comprehensive score of students' time to complete training tasks and correct tasks. This provides a better assessment of students' ability to solve programming tasks. First calculate the student's learning efficiency $e = \log^{S/t}$. Here, S is the student's score for all programming tasks and t is the time the student takes to complete all programming tasks. Therefore, the formula for calculating the efficiency score is as follow:

$$E = \frac{e}{e_{\max}} \times s. \quad \textbf{(3)}$$

As above, E is the efficiency score of students, $e_{\max}$ is the highest learning efficiency of students in the class, and s is the corresponding score. However, if the student does not complete the corresponding programming task, the student's efficiency score on the programming task will be 0. In the experiment, we set an efficiency score of 5 points for each level.

6 Experimental Verification

Constructing and verifying programming test questions is a complex task, which requires reasonable programming questions to be distributed and analyzed and verified with course scores. In this paper, we implement this process concretely. We designed a test paper on a programming topic, and then organized the third-year students to take the test on this topic. We assigned a specific comprehensive score for the students' test results. At the same time, the test paper was also distributed to some senior students to complete, and the test results of senior students were scored. Then, a comparison test was conducted between the test results of juniors and seniors. Using the scores of the test questions as a baseline of students' programming skills, the correlation between course scores and programming skills was analyzed.

6.1 Objective

Through our experiments, we aim to assess the correlation between course performance and programming skills. To the end, we propose three simple hypotheses. The first basic assumption is that the better the programming skills, the more correct they will be at solving the task. Because the better the programming skills and the richer the programming experience, the more experienced students participate in more programming projects than the inexperienced students, and thus can complete more programming tasks correctly. The second hypothesis is that experienced students are faster at analyzing problems and code because they have done more and more frequently in programming problems before, and they know how to better solve problems on their own. The third hypothesis is that students with better coding skills are also less evaluated (without other human factors) when completing coding tasks.

For the verification of these three hypotheses, we have integrated into specific experiments.

6.2 Material

We have designed five programming tasks for programming, which are two Java language programming tasks and three C language programming tasks. The score setting for each question is twenty points, including fifteen points for correct clearance and five points for efficiency. Among them, a certain difficulty level is also set between the questions. The Java language programming task is generally relatively simple. The third question of the C language programming task is the most difficult, and is regarded as a question with a degree of distinction.

We give students specific topics and some simple programming tips, let them complete these programming tasks, pass the evaluation in the EduCoder platform, and then count the students' completion. In Figure 3, we show the original topic and code of the first programming task. Since it is the first topic, the setting is relatively simple. The requirement of this topic is to count the total score of each student, and use the Java language to code a program.

```
/*
* Task: Calculate the average score of each person.
* Output style: total score of student number x: y
*
**/

public class PassWord {
    public static void main(String[] args) {
        int[][] arr = new
        int[][]{{90,88,87},{89,90,77},{66,78,60},{77,90,90},
        {8,9,78,67},{78,87,88}};
/********** Begin **********/
        int x,y;
        for( x=0;x<arr.length;x++){
            int sum=0;
            for(y=0;y<arr[x].length;y++){
                sum+=arr[x][y];
            }
        System.out.println(x+1+"student's total score: "+sum);
        }
/********** End **********/
    }
}
```

Figure 3. Source code of the first task

The remaining programming tasks are to use the Java language to output the student or teacher's score sequence, find the sum of the score sequence, insert the sort, and print the calendar. Printing the calendar has a certain complexity and can better distinguish students' programming abilities. There is also a certain inspection of the students' coding standards to test whether the students' code comments and function names are in compliance with the standards. We not only count the specific grades of middle school students on the EduCoder[1] platform, but also check the details of the experiments of students who have not passed the customs, and give them a specific score.

To match the average programming level of undergraduate students, we have selected some basic programming algorithms and problems. However, a question with a certain degree of difficulties was set up (this information can also be obtained in the small survey of the difficulty feedback of the students later), and the calendar of a certain month of the year was printed out. Some students with rich programming experience are also more aware of their own ideas and purposes to solve problems in actual programming projects. Thus, we expect that only students with a great deal of prior programming experience can do this task well.

There are five programming tasks. In the process of students completing the questions, the EduCoder platform will count the students' completion time, the number of completed questions, the completion time, the number of evaluations, and the efficiency of the questions, so that we can pass these data and give them a final score to distinguish their programming skills.

6.3 Subjects

All the research subjects were college students from the school. There were 124 main research subjects, all of whom were juniors. In addition, five senior students were tested on the same programming task, and a test result was obtained. Among these research objects, due to the change of professional training direction, the third-year students had learned two programming languages, C and Java at the same time, while the fourth-year students had only learned the C language. The other courses were basically the same.

6.4 Execution

The experiment was completed in April 2022. The students did not know the specific purpose and thought they were just participating in a lab class. Then, we briefly introduced the topics and some experimental requirements to the students. During the whole experiment, the students could only complete the programming tasks independently. They could not read textbooks or materials, and could not discuss the answers with others. In addition, in the process of students completing programming tasks, the front programming tasks were closed and submitted after a period of time, so that the experimental data could have a better authenticity. At the end of the time, we also conducted a return visit to some students to understand their views on these topics, whether the work was completed as planned during the experiment, and the difficulty level of the topics.

Five seniors also took the same test at another time, completing three of the programming tasks independently. Since they knew the purpose of the experiment and participated voluntarily, the experiment was improved.

1 https://www.educoder.net/

6.5 Deviation

When arranging for these students to participate in the experiment, due to time constraints, we failed to accurately distribute the 5 questions according to the correct difficulty level when we released the tasks in the EduCoder platform. As a result, the number of students completing the later programming tasks was smaller. In addition, when setting the task, another training task was not set for the last programming question, so the data of the two programming questions are combined.

7 Experimental Result

Before beginning this experiment, we propose three research questions:

RQ1. Can the programming skills of novice programmers be measured by course scores?

RQ2. If yes, which courses can measure programming skills?

RQ3. How does the assessment content, assessment methods and process management of the course affect the results?

From the experimental results, we can conclude that with the difficulty of the programming task and the answering time, the students' answer scores are also different. Relatively speaking, the simple and advanced tasks have more correct numbers. The last harder programming task was done correctly by fewer people. Then, after correlating all students' programming proficiency test scores with their course scores, it was found that Software Engineering course scores are the most correlated with programming test scores, and this score has a higher correlation with College English (1) scores and is very unrelated to Humanities and Social Sciences. The three research questions are specifically answered below.

7.1 Means and Standard Deviations

In Table 1, we outline how students completed these programming tasks, where the column average represents the average time in minutes. Because not all students completed these tasks, it is necessary to count their number of completions and the number of people who completed them correctly. As can be seen from the table, the first three tasks took less time and more people completed them correctly. Task 4 took the longest and had the largest number of participants, but the least number of people completed it. When setting tasks, the first few tasks will close the submission entry in turn after a period of time. Only task 4 is set at the end, and task 4 is also the most difficult task.

Take into account the premature closing of the submission entry during the experiment, and a comprehensive examination of the students' programming skills. We also opened the specific experimental situation of those students who participated in the task but failed to complete it correctly, and gave a comprehensive score to their answers. Particularly in the last task, by looking at the students' answers, we assigned a comprehensive score according to the reference answers, as well as the requirements of the coding standard.

Table 1. Overview of completion of each task

Variable	Mean	N	Correct
Task 1	10.94	81	71
Task 2	8.57	56	22
Task 3	16.07	92	53
Task 4	32.90	111	13

Mean: Average time for students to complete this task;
N: Number of students who complete this task;
Correct: Number of students with correct solution.

7.2 Correlations

We collated the test data of 124 students and then performed a correlation analysis among all students' coding proficiency test scores and their course scores. Since we are not sure whether the relationship is linear, we used the Spearman rank correlation. We obtained a preliminary result: among these courses, the correlation of test scores and Software Engineering is the strongest, and it is also closely related to Scientific Computing Language, Database Principles and Applications I, Data Structure, Operating System, Java Programming, and Computer Composition and Structure, and extremely irrelevant to Humanities and Social Sciences. Of the 13 courses, 7 courses are most significantly relevant. The details are shown in Table 2, which gives the Spearman rank correlation between test scores and course scores.

Table 2. The Spearman rank correlation between test scores and course scores

No.	Course	ρ	N
1	Software Engineering	.454**	124
2	Scientific Computing Language	.334**	124
3	Database Principles and Applications I	.297**	124
4	Data Structure	.273**	124
5	Operating System	.265**	124
6	Computer Composition and Structure	.254**	124
7	Java Programming	.232**	124
8	Big Data Storage and Processing	.222	124
9	Digital Image Processing II	.197	124
10	Advanced Language Programming	.183	124
11	College English (1)	.179	124
12	Humanities and Social Sciences	.117	124
13	Computational Methods	.078	124

7.3 Control Experiment

We also make a simple comparison and analysis of the experimental data of five senior students and the experimental data of junior students. Primarily comparing their scores for task 3 and task 4 in their programming tasks, it is clear that seniors perform better and have higher scores than juniors. As shown in Table 3, the average scores of seniors are significantly higher than those of juniors.

Table 3. Comparison of average scores

Variable	Juniors	Seniors
Task 3	7.82	19.06
Task 4	15.97	20.00

8 Exploratory Analysis

Because only the course scores and programming test scores of juniors are analyzed, the data is relatively small and single. An analysis was then conducted on the course scores of 129 senior students. We extracted their scores from two representative courses, Data Structures and Advanced Language Programming, and performed a simple score merging. We also sorted the scores from high to low, as one of their programming scores. Then, we organized five of these students to rank the programming skills of their seniors in their majors. We compared the ranking results given by them with the ranking results of the scores of the two courses, and found that nearly half of the students who ranked in the top 20 in programming skills were in line. Therefore, we decided to use these two course scores as a reference for programming scores, and performed a Spearman correlation analysis with the remaining course scores.

The obtained results show that Software Engineering, Database Principle and Application I, Operating System, Computer Composition and Structure, and Scientific Computing Language have the strongest correlation. This result is close to the previous analysis results of the third-year students, which also shows a reliability of the experimental results.

We also had a brief panel with these five graduating seniors, mainly to find out what they think can improve the programming skills of novice programmers. Their main point of view was that to improve their programming skills, they need to practice more programming problems and look at example codes for reference. If there is a problem, they can look for some suitable solutions on the Internet, such as the bilibili[2] website. It is important to do programming questions and algorithm questions, so that programming skills can be truly improved. In addition, they also gave their own opinions on the completed test questions. They believed that the earlier questions were relatively basic questions, and the last question had a certain degree of difficulty but was more interesting and novel.

9 Discussion

The EduCoder platform used in the experiment is an online practical teaching platform. It is an online practical teaching service platform and innovative environment widely used in domestic universities. On the EduCoder platform, we set up five programming tasks, including two Java language programming questions and three C language

2 https://www.bilibili.com/

programming questions. It is because Java language and C language represent process programming and object-oriented programming respectively, which can measure students' programming skills more comprehensively. The platform can set the time and scoring rules for students to complete tasks. It can also check the student's code to judge the situation of a student's work. There is also a code quality score, but because of its imperfect scoring rules, we finally use manual scoring to complete this operation.

In the process of setting programming tasks, we did not think of a factor in the order of students' problem-solving. Therefore, we did not correctly sort the levels according to the difficulty level when setting the levels, resulting in insufficient participation of students in some of the simpler tasks. The questions in this part were extracted, the students' answers were analyzed in detail, and they were given appropriate marks to balance the overall data.

10 Conclusion

Programming skills are important for novice programmers. Therefore, we urgently need a means or standard that can measure their programming skills. In general, programming skills are difficult to define accurately. Therefore, it threatens the validity of the experiment and makes the entire experiment more difficult to interpret.

Through Spearman correlation analysis of junior students' test scores and course scores, it was found that junior students' software engineering courses have the strongest correlation with programming skills, and there are another 6 courses as a course group is highly correlated. Among them, college English also has a certain correlation with programming skills, indicating that English ability is very important for novice programmers. It is because that in some mainstream programming languages and programming environments, some English terminology words often appear, and programmers with higher English proficiency are more likely to master these terminology words and understand error statements. Comparing the test results of seniors and juniors, these novice programmers with better programming skills are faster and more accurate. We also found that, compared with other courses, the grades of courses that implement the OBE mechanism can better reflect the programming skills of novice programmers and measure their programming skills.

To improve the programming skills of novice programmers is important but not easy. In addition to more practice, it is necessary to establish a complete teaching system for these courses, and use their course scores to evaluate students' programming skills so as to better improve the programming skills of novice programmers.

In future work, we hope that more meaningful programming tasks can be set up in the experiment of programming test to obtain richer experimental data. In addition, it is necessary to further analyze the curriculum groups with significant correlation and explore the correlation between the curriculum group and measuring programming skills.

Acknowledgment

This work was supported by the Key Project of Anhui University Natural Science Foundation (Grant No. YJS20210453, KJ2020A0361, and KJ2021A1028), Anhui Province Scientific Research Planning Project (Grant No. 2022AH050953), National Natural Science Foundation of China (Grant No. 62002084, 61976005), University Synergy Innovation Program of Anhui Province (Grant No. GXXT-2022-047), and Key Project of Natural Science Research of Higher Education Institution of Anhui Province of China (Grant No. KJ2020A0363, KJ2021A1028).

References

[1] M. Zhang, *The Art of Code: Driving Software Development with Engineering Thinking*, Publishing House of Electronics Industry, 2022, ISBN: 9787121426711.

[2] S. K. Navandar, M. D. Laddha, A. W. Kiwelekar, Analyzing Performance in a Computer Programming Course Through a Two-System Model, *International Journal of Performability Engineering*, Vol. 18, No. 2, pp. 71-78, February, 2022.

[3] R. Bockmon, S. Cooper, J. Gratch, J. Zhang, M. Dorodchi, Can Students' Spatial Skills Predict Their Programming Abilities? *2020 ACM Conference on Innovation and Technology in Computer Science Education*, Trondheim, Norway, 2020, pp. 446-451.

[4] L. Zhao, X. Liu, C. Wang, Y.-S. Su, Effect of different mind mapping approaches on primary school students' computational thinking skills during visual programming learning, *Computers and Education*, Vol. 181, Article No. 104445, May, 2022.

[5] A. K. Erümit, Effects of different teaching approaches on programming skills, *Education and Information Technologies*, Vol. 25, No. 2, pp. 1013-1037, March, 2020.

[6] S. I. Malik, R. Mathew, A. Al-Sideiri, J. Jabbar, R. Al-Nuaimi, R. M. Tawafak, Enhancing problem-solving skills of novice programmers in an introductory programming course, *Computer Applications in Engineering Education*, Vol. 30, No. 1, pp. 174-194, January, 2022.

[7] S. K. Navandar, A. W. Kiwelekar, M. D. Laddha, The Impact of Cognitive Bias on Students' Programming Performance in an Introduction to Programming Course, *International Journal of Performability Engineering*, Vol. 18, No. 8, pp. 589-597, August, 2022.

[8] X. Zhang, J. D. Crabtree, M. G. Terwilliger, T. T. Redman, Assessing Students' Object-Oriented Programming Skills with Java: The "Department-Employee" Project, *Journal of Computer Information Systems*, Vol. 60, No. 3, pp. 274-286, 2020.

[9] S. Kleinschmager, S. Hanenberg, How to rate programming skills in programming experiments?: a preliminary, exploratory, study based on university marks, pretests, and self-estimation, *the 3rd ACM SIGPLAN workshop on Evaluation and usability of*

programming languages and tools, Portland, Oregon, USA, 2011, pp. 15-24.

[10] H. Y. Durak, The Effects of Using Different Tools in Programming Teaching of Secondary School Students on Engagement, Computational Thinking and Reflective Thinking Skills for Problem Solving, *Technology, Knowledge and Learning*, Vol. 25, No. 1, pp. 179-195, March, 2020.

[11] Y. Li, Y. Song, A. Moukrim, S. Yu, An Ability-oriented Approach for Teaching Programming Courses, *2020 IEEE Frontiers in Education Conference*, Uppsala, Sweden, 2020, pp. 1-6.

[12] Q. Sun, J. Wu, K. Liu, How are students' programming skills developed: an empirical study in an object-oriented course, *the 2019 ACM Turing Celebration Conference*, Chengdu, China, 2019, pp. 81:1-81:6.

[13] X. Yang, Integrated Teaching Content Design of Programming Courses Based On Ability of Algorithm Thinking and Program Application, *2021 4th International Conference on Information Systems and Computer Aided Education*, Dalian, China, 2021, pp. 759-761.

[14] D. Bilegjargal, N.-L. Hsueh, Understanding Students' Acceptance of Online Judge System in Programming Courses: A Structural Equation Modeling Approach, *IEEE Access*, Vol. 9, pp. 152606-152615, November, 2021.

[15] J. Siegmund, C. Kästner, J. Liebig, S. Apel, S. Hanenberg, Measuring and modeling programming experience, *Empirical Software Engineering*, Vol. 19, No. 5, pp. 1299-1334, October, 2014.

[16] V. C. Ahku, S. Panchoo, Implementing Personalised Learning For Mixed Ability Students For Computer Programming In A Learning Environment, *2019 Conference on Next Generation Computing Applications*, Mauritius, 2019, pp. 1-8.

[17] X. Wei, L. Lin, N. Meng, W. Tan, S.-C. Kong, Kinshuk, The effectiveness of partial pair programming on elementary school students' Computational Thinking skills and self-efficacy, *Computers and Education*, Vol. 160, Article No. 104023, January, 2021.

[18] M. Nakayama, M. Uto, M. Temperini, F. Sciarrone, Estimating Ability of Programming Skills using IRT based Peer Assessments, *2021 19th International Conference on Information Technology Based Higher Education and Training*, Sydney, Australia, 2021, pp. 1-6.

[19] M. Sagar, A. Gupta, R. Kaushal, Performance prediction and behavioral analysis of student programming ability, *2016 International Conference on Advances in Computing, Communications and Informatics*, Jaipur, India, 2016, pp. 1039-1045.

[20] F. Zaffalon, A. Prisco, R. d. Souza, D. Teixeira, M. Neves, J. L. Bez, N. Tonin, R. Penna, S. Botelho, Estimating the Multiple Skills of Students in Massive Programming Environments, *2021 IEEE Frontiers in Education Conference*, Lincoln, NE, USA, 2021, pp. 1-7.

[21] A. Prasad, K. Chaudhary, B. Sharma, Programming skills: Visualization, interaction, home language and problem solving, *Education and Information Technologies*, Vol. 27, No. 3, pp. 3197-3223, April, 2022.

[22] M. Tahaei, K. Vaniea, Recruiting Participants With Programming Skills: A Comparison of Four Crowdsourcing Platforms and a CS Student Mailing List, *the 2022 CHI Conference on Human Factors in Computing Systems*, New Orleans, LA, USA, 2022, pp. 590:1-590:15.

Biographies

Fanghui Zha received her B.S. degree in computer science and technology from Anhui Polytechnic University. She is currently studying for her M.S. degree in software engineering at Anhui Polytechnic University. Her current research directions include knowledge tracking and programming skills measurement.

Yong Wang received his B.S. and M.S. degrees in computer science from Anhui Polytechnic University, and he received his Ph.D. degree in computer science and technology from Nanjing University of Aeronautics and Astronautics. His current research interests include software testing, fault localization, and program debugging.

Liqiang Mao received his B.S. degree in computer science education from Anhui Normal University, and And he is currently employed in Wuhu Education Examination Center. His current research interests include software engineering, data analysis, computer education.

Jin Liu is currently pursuing undergraduate studies in Electronic Information Engineering at Chongqing University of Posts and Telecommunications. She participated in the compilation of Railway Communication Signal Technology and System Research and has won multiple awards. Currently, she is participating in Student Research Training Program on "Linear Prediction Analysis, Synthesis, and MATLAB Implementation of Speech Signals".

Xue Wang received the B.S. degrees in computer science and technology from Lijiang College of Guangxi Normal University. She is currently pursuing the M.S. degree in software engineering at Anhui Polytechnic University, China. Her current research interests include software testing, fault localization, and program debugging.