

Measuring Programming Ability for Novice Programmers

Xue Wang¹, Yong Wang^{1,2,4*}, Fei Yang³, Wenge Le¹, Shouhang Wang¹

¹ School of Computer and Information, Anhui Polytechnic University, China

² State Key Laboratory for Novel Software Technology, Nanjing University, China

³ Zhejiang Lab, China

⁴ Fujian Provincial Key Laboratory of Information Processing and Intelligent Control, Minjiang University, China

2200720102@stu.ahpu.edu.cn, yongwang@ahpu.edu.cn, yangf@zhejianglab.com,

2200220114@stu.ahpu.edu.cn, 2200220107@stu.ahpu.edu.cn

Abstract

Coding is a key activity in the software development process and a programmer's programming ability determines the software quality. Different from professional programmers, novice programmers usually refers to programmers who have learned a programming language for about three years. At this stage, measuring their programming ability is of great significance to improve their programming abilities. In previous work, researchers have proposed a variety of ways to measure programming ability for professional programmers. We set out to find out the best way to measure novice programming ability. We first exacted a questionnaire from published comprehension experiments for measuring programming ability. Then, we performed control experiments to compare the answers to the questionnaire with their performance. We found that module number and the number of programming-related websites visited seem to be a reliable way to measure programming ability for novice programmers. Furthermore, we perform exploratory factor analysis to generate a model to verify the effectiveness of our findings.

Keywords: Programming ability, Measurement, Novice programmers

1 Introduction

The software development process generally contains four activities: plan, design, coding and test. Programming is a necessary ability for programmers, and programming ability is helpful for the smooth progress of related work in software development. Different from professional programmers, novice programmers usually refers to programmers who have learned a programming language for about three years [12]. At this stage, measuring their programming ability is of great significance to improve their programming abilities. To meet the requirements of enterprises, more attention should be given to the programming ability of novice programmers.

From previous studies, programming ability was viewed as an important confound factor in software engineering experiments [1-2]. There is no consistent way to measure programming ability in academia. Researchers often use different ways to measure programming ability. The most

commonly ways can be divided into two categories; the first is to measure through programming tasks directly [3-5, 10-11, 13], and the second is to use assessment methods indirectly, including self-assessments [1-2, 6] and third-party assessments [7-9]. However, for the second way, researchers still experiment with programming tasks. Therefore, it can be seen that it seems to be a reliable method to measure programming tasks, but it is difficult to operate. It requires a certain amount of time and experimental environments. To resolve this problem, we propose to find out the best method to measure novice programming ability through a questionnaire survey. For practical reasons, we choose graduates as novice programmers in our control experiment.

To measure programming ability, the first thing is to find the best indicator. Therefore, we first exacted a questionnaire from published comprehension experiments for measuring programming ability. Then, we ask novice programmers to fill out a questionnaire that contained questions related to programming ability. Next, we collect

the scores of courses directly related to programming. Finally, we make a comparative analysis of the novice programmers' course scores and the answers to the questionnaire.

As result, we determine two variables as the best indicator for programming ability using correlation analysis and stepwise regression: the number of modules in the project and the number of programming-related websites visited. In addition, we hold a symposium, in which a list of people with top programming ability among the 104 subjects are cited for validation of results. Our main contributions are two-fold: (i) we provide a reusable questionnaire that contains common questions to measure programming ability; and (ii) we propose a new model to measure programming ability of novice programmers, which can provide suggestions and ideas for future researchers in measuring programming ability.

The content of this article is arranged as follows: Section 2 discusses the related work. In Section 3, a questionnaire on programming ability for novice programmers is presented. Section 4 discusses the inspection criteria of the questionnaire survey. Section 5 describes the experimental process and results in detail. Section 6 discusses the research results and Section 7 is the conclusion.

2 Related Work

To learn how researchers measure programming, we review the relevant literature. These literatures have proposed possible standards to measure programming ability, which can classify and predict the programming ability of subjects.

For example, Kleinschmager et al. assessed the impact of self-estimation, university marks, and pre-tests on subjects' programming experience [1]. They asked the subjects to carry out two programming experiments and compared the performance of the subjects in the experiment with self-estimation, university marks, and pre-tests. The study found that the self-estimation does not seem to be worse than university marks and predictive tests, and may even be better than the two.

In order to find a method to measure programming experience, Feigenspan and other researchers extracted questions for evaluating programming experience from published literature and compiled it into a questionnaire [2]. They compared the answers in the questionnaire with the subjects' performance in program comprehension tasks. Experiment shows that self-estimation seems to be an effective method to measure programming experience. The questionnaire is a common way of collecting data. For example, in literature [14], the authors also let subjects fill out the questionnaire before the experiment. This is related to our work. We also aim to find a questionnaire for novice programmers.

In addition, there are some papers that study the rationality of students as subjects [15, 17, 19]. For example, in order to study students' understanding of demand selection, Svahnberg et al. conducted a survey on demand selection [15]. Then, they compared student answers to data from previous industry practice studies. The result shows that there is a significant correlation between the views of students and those of professionals. Therefore, students as subjects have certain applicability in research.

3 Questionnaire

There are many methods for measuring the programming ability of subjects. Feigenspan et al. [2] have carried out a detailed review on this, mainly including years, education, self-estimation, unspecified questionnaire, size, unspecified pretest, and supervisor.

Based on these methods, this paper designs a questionnaire about programming ability, including the following questions: years, self-estimation, size, and education. In addition, we add problems related to programming ability found in practical learning. We hope to propose a more standardized questionnaire to provide more detailed indicators for measuring programming ability.

In Table 1, we summarize the questionnaire and show the specific questions of the programming questionnaire and how

subjects should answer them. Regarding the criteria for evaluation of the question, we first refer to the content of the questionnaire in [2], and then set it according to the opinions of experts and the actual learning situation of students, hoping to get more accurate answers about programming ability. In the column source, we summarize the problem into five categories. In the column abbreviation, the abbreviations of each question are given to facilitate the use of the rest of this article. Next, we give a detailed explanation of each question.

A. Years

This category is mainly use to examine the subjects' contact programming time. This should start with the time of subjects first programmed, including the first study of

programming grammar and writing hello-world-like programs. In the answer, we set four options from primary school to college. Generally speaking, the longer the subjects are exposed to programming, the more source codes they write, and the higher their programming ability.

B. Self-estimation

In this category, we ask the subjects to self-evaluate their programming ability. However, before this, we do not give a specific definition of programming ability as a reference, which requires subjects to make the corresponding self-estimation according to their intuitive understanding of programming ability. In addition, we ask the subjects to compare themselves with their classmates and professional programmers with 10 years of programming experience. In this way, subjects have a deeper understanding of their programming ability.

C. Education

This category includes the assessment of educational aspects. First, we ask the subjects how many data structure algorithms and programming languages they think they are

familiar with. No matter data structure algorithms or programming languages, we provide the most basic answers, and these are all learned in university. We believe

that the more algorithms and programming languages college students are familiar with, the higher their programming ability. Besides, we also ask subjects to list their favorite courses at university. Because the number of courses and their relevance to programming roughly show how much source code they implement. Thus, we get an indicator: the more the number of programming-related courses college students like, the more the programming ability they obtain during the course study.

D. Size

For these questions, we examine the subjects' project experience and the amount of code in the programming task. We believe that when subjects participate in project writing more code, their programming ability must be higher than those who write less code.

Table 1. Overview of questions to assess programming ability

Source	Question	Criteria for Evaluation	Abbreviation
Years	When did you start contacting programming?	Primary / Junior / Senior / University	s.Year
Self estimation	How do you evaluate your programming ability?	1: very inexperienced to 10: very experienced	s.PE
	How do you evaluate your programming ability compared with your classmates?	1: very inexperienced to 10: very experienced	s.Classmates
	How do you evaluate your programming ability	very	s.Experts

	compared with 10-year experienced programmers?	experienced 1: very inexperienced to 10: very experienced	
Education	What data structure algorithms do you think you are good at? Which programming languages are you familiar with? What are your favorite courses?	Linear-List / Stack and Queue / Tree and Binary / Sort and Search Graph Java /C, C++/Python /.NET PHP /Front-end development Programming language / Data Structure / Software engineering... (Not listed here in detail)	s.Algorithm s.NumLanguages s.NumCourses
Size	How much have you participated in programming projects with more than 1000 lines? How many modules (functions) do you have in your largest programming project? How many lines of code are included in the maximum programming project you participate in? How many lines do you have the total number of programming lines in four years of university?	no participants / 1 – 2 / 3 – 4 / 5 or more within 3 / 3 – 5 / 5 – 8 / more lines within 100 / 101 – 1000 / 1001 – 10000 / above 10000 within 1000 lines / 1001-10000 lines / 10001-50000 lines / more	p.NumCode p.NumModule p.MaxProject p.SumProject
Other	Do you like programming? Would you use your spare time to watch programming- related video resources? What websites have you visited related to programming?	Very like / Like / General / Dislike Yes / No MOOC / CSDN / GitHub / Script house / Blog Park	s.Favor s.Viedo s.NumSites

In addition, we add other questions to the questionnaire. First, we ask subjects how much they like programming, as we think interest can improve student programming ability. Secondly, we ask students about their study after class, whether they would take the initiative to learn programming-related online courses, and what programming-related websites they have visited. We aim at have a more detailed understanding of programming ability through this questionnaire survey.

4 Inspection Criteria

In this paper, according to the characteristics of novice programmers, we select some course scores as inspection criteria. We put forward the following assumptions:

Assumption 1. The programming ability of the novice programmers are related to the course group.

Assumption 2. The higher the score of novice programmers, the stronger their programming ability.

To this end, we collect the scores of computer science. By asking experts, we select courses related to programming ability from many courses, including: C, C++, Java, Data Structure, Java Web, and Python. The reason for selecting these courses is that these courses require students to write programs manually in the learning process. Therefore, we think that the scores of these courses can be used to explain

student programming ability. Next, we verify our assumptions by analyzing the selected courses.

First of all, we analyze the distribution of data. Figure 1 shows the relationship between the scores of courses and the number of students. The horizontal axis represents the selected courses, the vertical axis represents the number of students, and four gray legends represent different scores interval. We can find that the scores of examinations have an approximately normal distribution relationship with the number of students. This shows that the score is conducive to the identification and selection of students. Therefore, it can be used in the following data analysis.

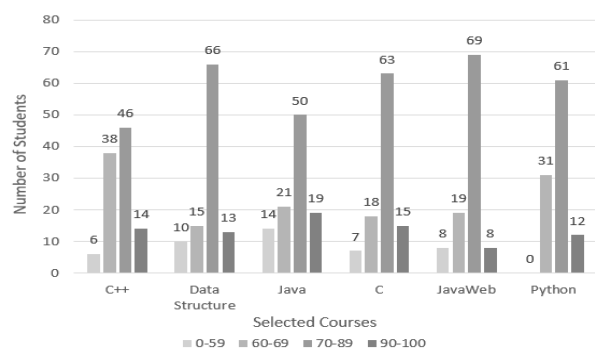


Figure 1. Scores of examinations

Furthermore, the relevance of each course in the course group to programming ability can be measured by the relevance of that course to other programming courses in the course group. To this end, we conduct a correlation analysis of these six courses. We choose Spearman rank correlation for data analysis. The reason is that Spearman rank correlation is mainly used to solve the correlation of ordered data, which can better measure the correlation strength and direction between two ordered variables [20], and the results are shown in the Table 2. We find that except C++ and Python, there is a significant correlation between all courses. For another thing, there is no obvious correlation between C++ and Python because of the huge differences in language style. But whether C++ or Python, there is a significant correlation between these two courses and other courses, indicating that these two courses must be related to programming ability. Therefore, we ignore the irrelevance between C++ and python and keep these two courses in the course group. Finally, we calculate the average score of each student in the course group and use it as inspection criteria for the questionnaire.

Table 2. Spearman correlations between courses

	C++	Data Structure	Java	C	Java Web	Python
C++	1.000	.387**	.407**	.557**	.399**	.138
Data Structure		1.000	.515**	.596**	.323**	.392**
Java			1.000	.500**	.359**	.364**
C				1.000	.424**	.419**
Java Web					1.000	.258**
Python						1.000

** : denote significant correlations($p < .01$)

However, for the criteria of verifying the answers to the questionnaire, it is not only necessary to ensure the rationality of curriculum selection, but also to test whether the scores can be used as criteria to measure programming ability through a third party. Therefore, in order to test whether the average scores of these six courses can be used as a measure of programming ability, we hold a symposium inviting some teachers and some students from the School of Computer and Information. During the symposium, we ask the teachers and some students about their impressions of the subjects' programming ability and ask them to make a list of people who are good at programming. After the meeting, we integrate a list of 26 students with some objectivity.

Finally, we sort the students according to their average score and compare them with the 26 students. We find that 14 of the top 20 students are on the list and 4 of the top 5 students are on the list. This is within a reasonable range, so we decide to use the average score as the criteria to verify the questionnaire answer.

5 Experiment

In this section, we describe the process and results of the experiment in detail. First, preprocess the answers to the questionnaire. Then, analyze the correlation between the answers to the questionnaire and the student's average course scores to find several factors that are significantly related to programming ability. Finally, other problems are excluded by stepwise regression to obtain the best indicator to measure programming ability.

5.1 Data Pre-processing

The data collected by questionnaire are mostly expressed in words and cannot be analyzed directly. Therefore, in this section we first preprocess the data. Mainly, we divide it into data digitization and outlier processing. Then, we give a brief overview of the answers to the pre-processed questionnaire.

5.1.1 Data Digitization

Most of the answers in the questionnaire appear in the form of options A, B, C, and D, so our pre-processing method is to replace A, B, C, and D in the data with 1,2,3,4. However, s. Algorithm, s. NumLanguages, s. NumCourses, s. NumSites in the questionnaire need another method for processing. Firstly, for the three questions of s. Algorithm, s. NumLanguages and s. NumSites, we give the same weight 1 to each option, and then the final answer is obtained by weighted summation. In addition, for the problem of s. NumCourses, we cannot simply give the same weight to each option, because the correlation between each course and programming ability is different.

Therefore, to solve this problem, we consult materials and ask experts, and finally reach a unified conclusion, that is, to assign different weights to each course according to its relevance to programming ability, and finally get the final answer by the weighted average. In Table 3, we show the corresponding weight values of each course.

Table 3. The weight value of courses

Course	Value
Programming Language	
Data Structure	3
Software Engineering	
Operating System	
Principles of Compilers	2
Mathematics courses	
Computer Organization and Design Fundamentals	1
Network Engineering	

The answers to other options include the following categories: Database, Front-end courses, Artificial intelligence big data courses, Back-end courses, Blockchain courses. Considering the correlation between these courses and programming ability, we give a weight of 2.


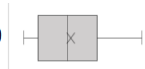

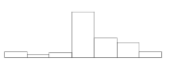

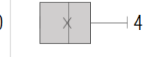
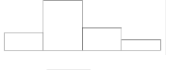
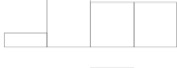

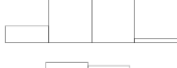



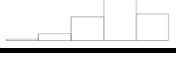
5.1.2 Outlier Processing

In this paper, we focus on data with logical errors. For example, on the three questions of self-estimation, comparative evaluation with classmates, and comparative evaluation with experts. We can find some obvious

phenomena that are not in line with common sense from the data collected in the questionnaire. These phenomena mainly include the same score of s.PE and s. Classmates and s. Expert, or s. Expert scored the same or higher as s. Classmates. We define this kind of data as abnormal data, because it accounts for a small proportion, so we directly delete records containing such data.

5.1.3 Overview

Table 4. Overview of answers in questionnaire

No	Question	Distribution	N
1	s.Year		104
2	s.Algorithm		104
3	s.PE		104
4	s.Classmates		104
5	s.Experts		104
6	s.NumLanguages		104
7	p.NumCode		104
8	p.NumModule		104
9	p.MaxProject		104
10	p.SumProject		104
11	s.Favor		104
12	s.NumCourses		104
13	s.Viedo		104
14	s.NumSites		104

In Table 4, we show the answers to the preprocessed questionnaire. Each image in the table represents the distribution of answer to each question. In order to represent the distribution of data more intuitively and avoid monotony for the overall beauty, we use different images. We find that most of the subjects start to learn programming from college, and only a few subjects start to learn from senior middle school. For self-estimation, most subjects give their programming ability a score of 6 or 7 points. However, when they are compared with professional programmers, they give themselves a lower score, such as 2 points, which is in line with our expectations. And for the project experience, we find that most subjects' code size is about 10000 lines, or within 10000-50000 lines. In general, there is a gap between the

programming ability of novice programmers and that of professional programmers.

5.2 Correlations

In Table 5, we give an overview of the correlation between the average score and the answers to the questionnaire. Here, we still use Spearman correlation analysis. We find that s.Classmates, p.NumModule, s.NumCourses and s.Numsites with subjects' average score have a significant correlation and the highest correlation with p.NumModule. For the remaining questions, we do not observe significant correlations.

Table 5. Spearman correlations of average scores with answers in questionnaire

No	Question	ρ	N
1	s.Year	-.013	104
2	s.Algorithm	.098	104
3	s.PE	.127	104
4	s.Classmates	.248*	104
5	s.Experts	.012	104
6	s.NumLanguages	.100	104
7	p.NumCode	.049	104
8	p.NumModule	.303**	104
9	p.MaxProject	.187	104
10	p.SumProject	.031	104
11	s.Favor	.110	104
12	s.NumCourses	.200*	104
13	s.Viedo	.057	104
14	s.NumSites	.214*	104

ρ : Spearman correlation.

N: number of subjects who completed this questionnaire.

For the completeness and accuracy of data analysis, we show the correlation between the score of each course and each question of our questionnaire in Table 6. We find that there are different degrees of correlation between each problem and each course. Of the 84 correlations, 16 are significant. For so many correlations, there is no meaningful conclusion without further exploration. But further research needs a large number of subjects, so we leave the analysis to work in the future. In the next section, we will continue to conduct the exploratory study.

Table 6. Spearman correlations of scores for each course with answers in questionnaire

	C++	Data Structure	Java	C	JavaWeb	Python
s.Year	-.107	.081	.018	.034	-.026	.061
s.Algorithm	.107	.098	.048	.106	.054	.042
s.PE	.129	-.058	.170	.211*	.197*	.041
s.Classmates	.151	.072	.304**	.249*	.209*	.156
s.Experts	.097	-.084	.076	.149	-.079	-.079
s.NumLanguages	.094	.114	.201*	.051	.068	.078
p.NumCode	-.009	-.074	.129	.119	.077	.059
p.NumModule	.181	.144	.284**	.332**	.198*	.252**
p.MaxProject	.152	.035	.258**	.236*	.072	.186
p.SumProject	-.006	-.110	.076	.124	.106	.054
s.Favor	.099	-.033	.118	.112	.127	.012
s.NumCourses	.023	.148	.089	.330**	.147	.066
s.Viedo	-.075	.022	.111	.037	.071	.036
s.NumSites	.236*	.312**	.068	.172	.205*	-.034

*: denote significant correlations ($p < .05$)

** : denote significant correlations ($p < .01$)

5.3 Exploratory Study

In this section, we will continue to explore the data and look forward to using stepwise regression to find the best indicator to measure programming ability. Stepwise regression is widely used in economics and psychology, but rarely used in software engineering [2]. Therefore, we first introduce the method of stepwise regression.

5.3.1 Method

Stepwise regression is a process of selecting variables in regression analysis. It introduces independent variables into the model step by step; if variables are statistically significant, they are included in the regression model and simultaneously remove variables without statistical significance. In general, it selects the most important variable from a set of candidate variables to build a regression model.

First of all, why should we select a set of candidate variables from many questions? From Table 5, we can find that not all the problems are related to the average score of the course. The candidate variables we select should be at least moderately related to the average score of the course. Because of the Spearman correlation analysis, the factors with moderate correlation are usually considered to have a certain relationship with the dependent variable, which has the value of research, and those factors without correlation can be ignored.

Second, why can't all factors with medium correlation with the dependent variable be selected as the measurement indicator? If so, we ignore an important issue that these factors themselves may be interrelated, even highly relevant. Now, we assume that there is a correlation between the two selected independent variables. If we use these two independent variables to measure programming ability, we will calculate the common part of the two independent variables one more time when calculating, which will lead us to overestimate the

impact of variables on programming ability [18]. For example, in our experiment, we found that s. ClassMates had a significant correlation with p.NumModule through Spearman correlation analysis, and their correlation degree was 0.350.

Therefore, in order to reduce the impact of the correlation between the independent variables on the problem, we use the stepwise regression method to explore [16]. To better introduce independent variables to the model, we should input independent variables with significant influence into the regression model from large to small according to the correlation between independent variables and dependent variables. For example, in this paper, we should first consider the problem with the highest correlation with the average score of the curriculum, which is p. NumModule. Then, consider the problem of high correlation, which is s.Classmates. If the existing independent variables in the model are no longer significant due to the introduction of the current independent variables, they are eliminated. The same is true for the rest of the problem, which is a repeated process. Through stepwise regression, we build a regression model so that in the end only significant variables are included in the model.

5.3.2 Results and Interpretation

In Table 7, we show the results of stepwise regression (specifically, we use the step method). By stepwise regression, we extract two questions: the number of

Table 7. Model results of stepwise regression

Question	Beta	t	p
p.NumModule	.289	3.092	.003
s.NumSites	.230	2.461	.016

modules in the project (p.NumModule) and the number of programming-related websites visited (s.NumSites). The beta value in the table shows the impact of variables on the average score of the courses. The higher the beta value, the greater the

impact. The model is significant ($p < .05$), and the adjusted R^2 is 0.15, which means that the explanatory rate of the extracted two questions to the average course score is 15%.

Therefore, the result of the stepwise regression is that the number of modules in the project and the number of websites related to browsing programming contribute the most to the average score of the course. That is to say, the more modules college students participate in projects, and the more frequently they browse programming-related websites, the higher their average scores. In addition, the factor extracted here is the number of modules in the project rather than the total number of lines of code, because modular programming can better reflect the programmer's logical thinking during programming and promote the improvement of programming ability. Why are these correlated? First of all, we are clear that the subjects of our survey are college students, who have relatively weak programming ability compared to professional programmers. If they are willing to look up learning and programming-related websites after class, then such students must have a certain interest in programming and are willing to spend time learning to program, so their programming ability must not be bad.

In general, we eliminate other problems in the questionnaire by stepwise regression, and finally determine the two problems of p . NumModule and s . NumSites as the best indicators to measure programming ability.

5.3.3 Experimental Verification

A regression model is obtained by a stepwise regression algorithm. The model gives the beta value of the question, which represents the weight of each question. We can calculate the value of students' programming ability in this model. We can set the programming ability as θ , the number of project modules as α , and the number of programming-related websites visited as β ; then, the formula for calculating the programming ability is (for simplicity, we ignore the constant term of the equation):

$$\theta = \alpha \times 0.289 + \beta \times 0.230 \quad (1)$$

Therefore, we can use this formula to calculate the programming ability of all subjects and sort them and then compare them with 26 students obtained from the symposium. The result shows that 11 of the top 20 students are on the list.

6 Discussion

6.1 Implication

The main work of this paper is to systematically analyze the relationship between novices' programming ability, relevant course scores, and other factors. By building a model to evaluate programming ability, the significant factors are analyzed. The result shows that p . NumModule and s . NumSites have the greatest influence on novices' programming ability. Although our research conclusion cannot provide a perfect explanation for the programming ability of novice programmers, it provides a reference. Whether it is for business managers or university teachers, they can combine our research conclusions with the actual situation to assist and guide their work. For example, the relevant conclusions can provide references for colleges and

universities to formulate training programs for computer professionals, and can also provide a basis for companies to evaluate graduates' programming ability during recruitment.

6.2 Threats to Validity

The first threat to validity is inspection criteria. For different inspection criteria, the conclusions may be different. In this paper, according to the characteristics of research objects, we select the course scores related to programming to represent the programming level of novices. But we need to consider how much the score of those courses can represent the programming ability. If the content of examination related to programming is less in proportion to the overall content, then the score of this course cannot represent the programming ability. In this paper, we examine the rationality of selected course scores by asking experts and teachers. Therefore, we believe we controlled this threat, and the inspection criteria selected in this paper is suitable for our research purpose.

The second threat is the sample size. Due to the limitation of practical reasons, we only can collect about 100 pieces of data. Although there are only about 100 pieces of data, we still consider using the existing mature methods to analyze the data to obtain more reliable conclusions. Another threat is the sample selection: we only select graduates as research objects. Our conclusions are only applicable to novice programmers with similar backgrounds, because our questions may have different meanings for novice programmers who have participated in the work. For example, s . Classmates is not suitable for novice programmers who are already working. For them, it should be compared with colleagues. When applying the results to novice programmers at work, other indicators, such as total code may be a better indicator. In the future work, we plan to expand our sample size and invite more different types of novice programmers to join our research, so as to obtain more diverse data to reduce these threats.

In addition, the fourth threat is the subjectivity of data. In the model of evaluating programming ability, we use highly subjective data such as questionnaires. Since there

is no good method to verify whether the answers to the questionnaire are in line with the actual situation, we ignore the evaluation of the objectivity of the questionnaire results and the screening of effective content in the current analysis because it requires a lot of support work. But, we still consider using other methods to deal with the subjectivity between self-assessment answers. Therefore, we think that we have made a certain effort to control this threat. In future research, we plan to confirm and screen the results of the questionnaire to improve the reliability of the results.

7 Conclusion

In this paper, we mainly focus on the programming ability of novice programmers and explore the significant factors affecting their programming ability. In order to carry out the research smoothly, we ask novice programmers to complete a questionnaire. We compare and analyze the answers of the questionnaire and their course scores, and construct a linear regression model to measure programming ability by stepwise regression method. The model shows that the significant factors affecting programming ability are the number of modules in the project (p . NumModule) and the number of

programming-related websites visited (s.NumSites). This conclusion can not only provide some learning methods for novice programmers to improve their programming ability, but also provide reference for enterprises to recruit employees and universities to train computer talents. In addition, this is a preliminary study and more work is required to test and improve this model. In future work, we plan to expand our sample size and invite more different types of novice programmers to join this study. We also need to confirm and screen the answers of the questionnaire to reduce the influence of the subjectivity of the questionnaire on the experiment and further improve the quality of our measurement.

Acknowledgements

This work was supported by the Anhui Natural Science Foundation (Grant No. 1908085MF183), Project 61976005, 61772270 supported by NSFC of China, Training Program for Young and Middle-aged Top Talents of Anhui Polytechnic University (Grant No. 201812), Natural Science Foundation of Zhejiang Province (Grant No. LQ21F020004, State Key Laboratory for Novel Software Technology(Nanjing University) Research Program (Grant No. KFKT2019B23), the Open Research Fund of Anhui Key Laboratory of Detection Technology and Energy Saving Devices, Anhui Polytechnic University(Grant No. DTESD2020B03), Anhui Province Quality Engineering Teaching and Research Key Project (Grant No. 2019jyxm0216, 2018jyxm0025) and Open Fund Project of Fujian Provincial Key Laboratory of Information Processing and Intelligent Control (Minjiang University) (Grant No. MJUKF-IPIC202109).

References

- [1] S. Kleinschmager, S. Hanenberg, How to rate programming skills in programming experiments?: a preliminary, exploratory, study based on university marks, pretests, and self-estimation, *Conference on Systems, Programming, and Applications: Software for Humanity*, Portland, Oregon, USA, 2011, pp. 15-24.
- [2] J. Siegmund, C. Kastner, J. Liebig, S. Apel, S. Hanenberg, Measuring and modeling programming experience, *Empirical Software Engineering*, Vol. 19, No. 5, pp. 1299-1334, October, 2014.
- [3] T. Effenberger, R. Pelanek, Measuring students' performance on programming tasks, *Sixth (2019) ACM Conference on Learning @ Scale*, Chicago, Illinois, USA, 2019, pp. 1-4.
- [4] J. Kittur, Measuring the programming self-efficacy of electrical and electronics engineering students, *IEEE Transactions on Education*, Vol. 63, No. 3, pp. 216-223, August, 2020.
- [5] S. Biffl, W. Grossmann, Evaluating the accuracy of defect estimation models based on inspection data from two inspection cycles, *International Conference on Software Engineering*, Toronto, Canada, 2001, pp.145-154.
- [6] C. Bunse, Using patterns for the refinement and translation of UML models: A controlled experiment, *Empirical Software Engineering*, Vol. 11, No. 2, pp. 227-267, June, 2006.
- [7] E. Arisholm, H. Gallis, T. Dyba, D. I. K. Sjøberg, Evaluating pair programming with respect to system complexity and programmer expertise, *IEEE Transactions on Software Engineering*, Vol. 33, No. 2, pp. 65-86, February, 2007.
- [8] J. Carver, L. Hochstein, J. Oslin, Identifying programmer ability using peer evaluation: An exploratory study, *Object-oriented Programming, Systems, Languages, and Applications*, Orlando, Florida, 2009, pp. 1-8.
- [9] J. E. Hannay, E. Arisholm, H. Engvik, D. I. K. Sjøberg, Effects of personality on pair programming, *IEEE Transactions on Software Engineering*, Vol. 36, No. 1, pp. 61-80, January-February, 2010.
- [10] R. Bockmon, S. Cooper, J. Gratch, J. Zhang, M. Dorodchi, Can students' spatial skills predict their programming abilities?, *Innovation and Technology in Computer Science Education*, Trondheim, Norway, 2020, pp. 446-451.
- [11] R. Bockmon, S. Cooper, W. Koperski, J. Gratch, S. Sorby, M. Dorodchi, A CS1 spatial skills intervention and the impact on introductory programming abilities, *The 51st ACM Technical Symposium on Computer Science Education*, Portland, Oregon, USA, 2020, pp. 766-772.
- [12] R. Lister, On the cognitive development of the novice programmer: and the development of a computing education researcher, *the 9th Computer Science Education Research Conference*, Leiden, Netherlands, 2020, pp. 1-15.
- [13] G. R. Bergersen, D. I. K. Sjøberg T. Dybå, Construction and validation of an instrument for measuring programming skill, *IEEE Transactions on Software Engineering*, Vol. 40, No. 12, pp. 1163-1184, December, 2014.
- [14] H. Erdogmus, M. Morisio, M. Torchiano, On the effectiveness of the test-first approach to programming, *IEEE Transactions on Software Engineering*, Vol. 31, No. 3, pp. 226-237, March, 2005.
- [15] M. Svahnberg, A. Aurum, C. Wohlin, Using students as subjects-an empirical evaluation, *2008 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, Kaiserslautern, Germany, 2008, pp. 288-290.
- [16] M. S. Lewis-Beck, *Applied regression: An introduction*, Beverly Hills Calif, 1980.
- [17] M. Host, B. Regnell, C. Wohlin, Using students as subjects-a comparative study of students and professionals in lead-time impact assessment, *Empirical Software Engineering*, Vol. 5, No. 3, pp. 201-214, November, 2000.
- [18] J. Cohen, P. Cohen, *Applied multiple regression/correlation analysis for the behavioral sciences*, Addison Wesley, 1983.
- [19] D. Boehm-Davis, L. Ross, *Approaches to structuring the software development process*, General Electric Co Arlington VA Data Information Systems, October, 1984.
- [20] M. Carpenter, The new statistical analysis of data, *Technometrics*, Vol. 42, No. 2, pp. 205-206, May, 2000.

Biographies



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.



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.



Fei Yang received his B.S. and M.S. degrees in computer science from Shanghai Jiao Tong University, and he received his Ph.D. degree in computer science from Eindhoven University of Technology. His current research interests include deep learning, machine learning systems, and concurrency theory.



Wenge Le received the B.S. degrees in automation from Huangshan university. He is currently pursuing the M.S. degree at Anhui Polytechnic University, China. His current research interests include software testing, fault localization, and program debugging.



Shouhang Wang received the B.S. degrees in electrical engineering and automation from Xuzhou Vocational College of Industrial Technology. He is currently pursuing the M.S. degree at Anhui Polytechnic University, China. His current research interests include software testing, fault localization, and program debugging.