B-Learning on Novice Programmer Learning with Roles of Variables

SHI Nianfeng¹², ZHANG Ping¹,* and SUN Ximing²

¹ Luoyang Institute of Science and Technology, 471023 Luoyang, China
² the University of Southern Queensland, 4350 QLD, Australia
³ Henan University of Science and Technology, 471023 Luoyang, China
* zhangping76@126.com

Abstract — A role-based b-learning model is proposed to teach novice programmers learning C language programming by applying the roles of variables to the b-learning teaching method. The effectivenss is evaluated by the Structure of Observed Learning Outcomes (SOLO) taxonomy. The participants of the research were divided into an experimental group and a control group. The students of the control group learned programming with the classical b-learning method. The students in the experimental group learned programming using the b-learning approach with variables visualization to provide scaffolding. For the purposes of determining the effects of the role-based b-learning teaching method, the SOLO levels of code reading and writing were respectively graded according to the SOLO categories for program comprehension and construction. Data analyses show that there is a higher level of performance and cognitive ability in the experimental group than the control group. This result suggests that combining b-learning with the role of variable is effective on improving novice programming skills.

Keywords- B-learning; novice programmer; roles of variable; SOLO taxonomy; teaching/learning strategies

I. INTRODUCTION

Learning computer programming is difficult and problematic for many novice programmers [1, 2]. At the Luoyang Institute of Science and Technology (LIT), approximately 26.4% of computer science students failed the advanced programming subject in the first semester from 2007 to 2014. One explanation is that, most of the novice programmers struggle to perform relational reasoning [1, 3]. Previous studies showed that the engagement with relational reasoning explicitly through such as reading and explaining could significantly improve novices’ program skills [4, 5].

Regarding the most effective way to teach novices reading and explaining the code, it is believe that b-learning is a promising technique. ICTs have become major sources and basis for learning in higher education, which can provide educators an environment to place their online course materials and for students to receive that education while interacting with other students/teachers outside the class setting [6]. B-learning, otherwise called blended learning, is a teaching and learning approach combining traditional face-to-face instruction (F2F) and e-learning. Program learning with the b-learning teaching method, teachers can provide novices with lots of program cases to reading and explaining in the web-based learning platform. However, it is big challenge for novices to read or explain a program. Novices typically lack of the experiences and theoretical knowledge of computer program. As a result, they are sometimes either unable to grasp the syntax and semantics of a particular programming language [7] or unable to read and comprehend program codes [8]. Therefore, b-learning places a cognitive load on the novice programmer and it is very important to decrease extraneous cognitive load and ensure the educational technology be fully devoted to learning [9].

It seems that the concept of roles of variables present the b-learning on novice programmers learning. Previous studies showed that roles of variables enabled novice programmers to mentally process programs in a comprehensive way [10].

The purpose of this study is to test whether roles of variables can provide scaffolding to improve the pedagogical effectivenss of b-learning on novice programmers learning. The program learning is a cognitive process, and SOLO taxonomy is a popular choice to evaluate student ability to read a program and ability to write one [11, 12]. Thus, in this article, the SOLO taxonomy was used as an evaluation method.

II. LITERATURE REVIEW

A. Overview of B-Learning

B-learning, is a fusion of e-learning and the traditional F2F room learning. According to [6], the basic principle of b-learning is that F2F oral communication and online written communication are optimally integrated such that the strengths of each are blended into a unique learning experience congruent with the context and intended educational purpose. Previous studies have shown that b-learning can be used to increase students’ engagements, promote a shift from a more teacher-centered approach to learner-centered focus, and encourage peer-to-peer learning [13].

B-learning has been successfully used in computer programming course. B-learning environment on program learning can resulted in marked improvements in pass rates and positive student evaluations [14]. However, according to [15], just blending face-to-face learning with information...
B. Roles of variables

Variables play an important role in the program, but novices thought that the concept of the variable was too abstract to grasp and argued that it took them too much time and effort to resolve logical errors caused by misuse of variables [2]. To improve learning computer programming, the concept of the roles of the variables was introduced in [16]. A role was characterized by the sequence of successive values of a variable and the dependence of the variable on other variables, rather than the way that the variable is used [17]. According to [18], there are eleven roles of variables in novice-level C language programming and their informal definition, including fixed value, stepper, most-recent holder, most-wanted holder, gatherer, follower, one-way flag, temporary, organize, container and walker.

Prior studies showed that roles of variables could improve novices’ code comprehension and debugging skills [10, 17, 18], help them in building more general and abstract understanding of the program structures and the relation to problem domain concepts [19]. Students with the role-based teaching tended to stress deep program structures that were a sign of better comprehension and had fewer problems with variables in program construction[18]. However, their research showed insufficient evidence to conclude that the role-based teaching improved students’ performance on program construction.

Roles of variables are cognitive concepts rather than technical. To read or write a program, students need a holistic point of view as well as a local perspective. Using variables often requires a relatively local perspective, while understanding and writing a program requires a holistic approach in order to build relation and multi-structure in a program. To assess students’ ability in these two dimensions of the effect of the role-based teaching method, SOLO taxonomy was adopted.

C. SOLO taxonomy

SOLO (Structure of Observed Learning Outcomes) taxonomy offers a way of describing the growing complexity and hierarchy of a learner’s activity: being able to look at only one tree, seeing a few isolated trees, or seeing a whole forest. SOLO taxonomy, based on the amount of the learner knows (quantitative) and the integration of the detail into a structural pattern (qualitative), describes the growing complexity of a learner’s activity. By using this taxonomy, students’ answers can be classified according to the level of integration that they demonstrate rather than absolutely correctness [20].

Regarding to evaluate students’ program skills, both reading and writing ability of students needs to be assessed. This is because there is a wide consensus that code reading and code writing are related activities and each may help the other [21]. However, this relation is not strict hierarchy, and they develop in parallel. According to [22], SOLO evaluation methods for code reading questions and code writing tasks are also different. In this research, the SOLO categories for program comprehension are used for evaluating program reading skill and SOLO categories for construction were used for program writing skill.

SOLO categories for program comprehension proposed by [20] were used to evaluate the program reading and understanding ability of the student because previous studies have shown SOLO can be used to reliably classify code the student responses to comprehension questions [11, 23]. This is because 1) SOLO taxonomy is consistent with the student’s performance in program comprehension [23] and 2) SOLO taxonomy can help teachers examine whether students had the ability to read several lines of code and integrate them into a coherent structure[11].

SOLO categories for construction proposed by [22] was used to evaluate the writing ability of the student. An initial set of guidelines and descriptors for using SOLO to classify student code writing skills was proposed by [20]. However, the process of mapping from student code to the SOLO taxonomy was very context bound and question specific [12, 23], therefore, there were many of the ambiguities in the use of these guidelines to assess student code writing tasks. To solve this problem, a refined quality evaluation framework based on ground theory was proposed in [22], which can help teachers to map novices’ responses to program construction into SOLO categorizations through identifying critical elements. This framework has been proved to be a more consistent and supportable SOLO categorizations for novice programmers’ responses to program construction [12].

III. RESEARCH METHOD

A. Participants

The participants of this study consisted of 57 first-grade undergraduate students of computer science of LIT in the C Language Course for the second semester of the 2013/2014 session. All students were split into control group and experiment group based on the two intact classes, and were randomly assigned as control and experimental group by the researchers. The experimental group consisted of 27 students and the control group 30.

No sample was taken since the course was an obligatory one.

B. Instruments

1) Learning context

According to the 2013 version of the course, C Language Course of LIT lasts for thirty-six lessons in eighteen weeks. The courses in experimental and control groups were conducted using in the b-learning model (see Figure 1). Figure 1 illustrates the educational activities in the experimental group and the control group. Students from the control group (b-learning group) learned programming with the classical b-learning teaching approach. A revised b-learning context through roles of variables as scaffolding was designed for the experimental group (role-based group).
In Figure 1, the F2F learning is a traditional case-based teaching on C programming learning focusing on theoretical study, while the web-based learning is a classical e-learning focusing on the program reading and writing. The details of F2F learning and web-based learning in the b-learning context are depicted as followings:

a) F2F learning. F2F learning is a programming language instruction with the classic case-based teaching method except the instruction of roles of variables (see Figure 1) in the experimental group. Before the start of the session, some case programs were collected by the researchers and then made into handouts. During the session, the teacher conducted the course of the C Language Programming covered by the cases program.

b) Web-based learning. To implement the b-learning on novice programmer learning, two functions were conducted by the web-based platform. First, the web-based platform is a classical e-learning system. There are many pedagogical resources which can be accessed by novices, including project cases, homework assignments, slides, and test items bank. These pedagogical resources can be used to experience synchronous learning or personalized learning by novice programmers. Second, the web-site is a laboratory platform. The laboratory tasks, including program reading, program explaining and program tasking, were conducted through the web-based C language development interface (web-based IDE). To complete a program writing task, novices were asked to watch a case program video (b-learning group) or a case program with role-based animation (role-based group), and then, they constructed a new program by revised the case program. The web-based learning focuses on the program reading and explaining.

The educational activities of role-based teaching in the experimental group can be summarized as following. Firstly, the roles of variables were methodically presented in the F2F learning. In the beginning of the session, exact definitions of the above 11 roles with short program examples in C language were translated into Chinese, and printed as booklets. Roles were annotated in the case programs. At the beginning of the course, printed booklets and case programs were handed out to students. In the practice, the teacher introduced the definition, and presented students with the example program in the PlanAni system when a new role appeared. The PlanAni system is a tool supporting the concept of the roles of variable, where all variables of the role and their operations were visualized as the role images and animations. Meanwhile, in order to elaborate intuitively the life cycle of the variable and the function of the role, some code tracing and debugging technologies were employed. The tracing and debugging procedure can be summarized into 1) the teacher set breakpoints in the program code of the role and 2) the teacher debugged program systematically with the visual debugging tools of Visual C++ 6.0 IDE, such as QuickWatch, Set Next Statement, Step into and so on. In this process, the teacher and students worked together to predict the next outcomes of the variables, and to view the actual results. To ensure the students can thoroughly understand the roles of variables and
properly apply them to the programming. Participants must make out the role and reason for using it in the next two homework programs when the teacher finished instructing a role.

Secondly, roles of variables were systematically practiced in web-based learning. At the beginning of the laboratory task in the web-based IDE, novices were asked to watch a case program with role-based animation in a web-based role visualization module (see Figure 2), named as e-PlanAni, which was web browser plugin with the same function as the PlanAni system. At the end of the laboratory task, novices were required to list all roles used in their programs and explain them function.

![Figure 2. The interface of the e-PlanAni system.](image-url)

2) **Final examination**

The final examination consisted of three types of questions similar to what adopted by [18]:

- Basic knowledge measurement: True or false question (TFQ) and multiple-choice question (MCQ). TFQs concerned the basics of C language programming, e.g., a definition of integer data or statement of integer assignment operation. MCQs concerned the basic knowledges or skills of C language programming, e.g., deciphering the output of three lines long program.
- Program comprehension (COMP): Participants were presented with three programs.
- Program construction (CONS): Participants were asked to write code that requires them to initially use a group of integers input from a user as the elements of integer array, and to arrange the array in descending order. This is followed by asking participants to insert a new integer into the array in descending order.

C. **Experiment procedure**

C language course was an eighteen-week offering, including the theoretical content learning, laboratory practice phase, and final exam. The research procedure started from the first week of the second semester of the 2013/2014 session.

Participants of the experimental and control groups were not informed that they were being involved in the research process. In the final examination, participants were asked to complete a two-hour paper-pencil test.

D. **Method of data collection**

To conduct the data analysis, two variable data sets were collected: the final paper-pencil marks, and SOLO levels of the code reading and the code writing. To assess the SOLO level of a student, the researchers first assigned the SOLO level to a participant’s answers to COMP and CONS according to the SOLO categories for the program comprehension and construction, respectively.

a) **SOLO categories for program comprehension**, which was proposed in [20].

- **Relational (R)**: A summary of what the code does in terms of its purpose (the forest).
- **Relational Error (RE)**: A summary of what the code does in terms of its purpose, but with some minor error.
- **Multistructural (M)**: A line-by-line description of all the code (the trees).
- **Other (O)**: Any other description of part or all of the code, displaying no real evidence of understanding of the code as a whole.

b) **Refined SOLO categories for program construction**, which was based on [22].

- **Relational (R)**: Provides a valid well-structured program that removes all redundancy and has a clear logical structure.
- **Multistructural (M)**: Represents a translation that is close to a direct translation. The code may have been reordered to make a more integrated and/or valid solution.
- **Unistructural (U)**: Represents a direct translation of the specifications. The code will be in the sequence of the specifications.
- **Prestructural (P)**: Substantially lacks knowledge of programming constructs or is unrelated to the question.

The SOLO level obtained according to SOLO categories was then converted to its corresponding scores, as shown in Table 1. This numeric form reflects the ordinarily of the original SOLO level.

<table>
<thead>
<tr>
<th>SOLO category</th>
<th>Converted Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program comprehension</td>
<td>Program construction</td>
</tr>
<tr>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>RE</td>
<td>M</td>
</tr>
<tr>
<td>M</td>
<td>U</td>
</tr>
<tr>
<td>O</td>
<td>P</td>
</tr>
</tbody>
</table>

The SOLO level obtained according to SOLO categories was then converted to its corresponding scores, as shown in Table 1. This numeric form reflects the ordinarily of the original SOLO level.
IV. RESULTS

An independent-samples t-test was conducted for participants’ final examination scores. The data in Table 2 showed a significant difference (t=2.61, df=55, p=0.012<0.05) in the final examination scores between experimental (N=27, M=79.7, SD=10.01) and control (N=30, M=71.8, SD=12.65) groups. The mean of final scores for the role-based group was 10% more than the b-learning group, indicating that the role-based teaching approach was more effective than the classical b-learning teaching method.

<table>
<thead>
<tr>
<th>Group</th>
<th>N</th>
<th>M</th>
<th>SD</th>
<th>t</th>
<th>df</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experimental</td>
<td>27</td>
<td>79.7</td>
<td>10.01</td>
<td>2.61</td>
<td>55</td>
<td>0.012</td>
</tr>
<tr>
<td>Control</td>
<td>30</td>
<td>71.8</td>
<td>12.65</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The researchers also examined whether the roles of variables had influenced novice programmers’ basics of C language programming learning. This was performed through testing the difference in participants answering to TFQs and MCQs questions in the final exam. The independent-samples t-test in Table 3 showed that there is no significant difference (t=0.14, df=55, p=0.888>0.05) between the experimental group (N=27, M=39.7, SD=4.23) and control (N=30, M=39.5, SD=6.15) group.

<table>
<thead>
<tr>
<th>Group</th>
<th>N</th>
<th>M</th>
<th>SD</th>
<th>t</th>
<th>df</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experimental</td>
<td>27</td>
<td>39.7</td>
<td>4.23</td>
<td>0.14</td>
<td>55</td>
<td>0.888</td>
</tr>
<tr>
<td>Control</td>
<td>30</td>
<td>39.5</td>
<td>6.15</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The researchers examined the SOLO level of the code reading and code writing of participants. As the Shapiro-Wilk Test showed participants’ scores of program comprehension and program construction in the experimental and control groups were not normally distributed, two independent Mann-Whitney U tests were used to test the difference between the two groups in the program comprehension and construction. Table 4 indicated that the SOLO score for program comprehension in the experimental (N=27, MR=34.24) group was significantly higher (U=263.5, p=0.019<0.05) than the control (N=30, MR=24.28) group. Similarly, Table 4 indicated that the SOLO score for program construction in the experimental (N=27, MR=34.67) group was significantly higher (U=252.0, p=0.011<0.05) than the control (N=30, MR=23.90) group. The results, as far as the code reading and code writing in C language were concerned, indicated a novice programmer in the role-based group had a higher cognitive level than the b-learning group. In other words, the role-based teaching was more effective than the classical b-learning teaching method.

The relationship between the score of the final exam and the SOLO level of the code reading and code writing were tested. Due to the non-normal distribution and monotonicity of the experimental data, the Spearman’s rank-order correlation was utilized. Table 5 showed two positive and significant correlations between the score of the final paper-pencil test and the score of SOLO in the program comprehension (r=0.799, p<0.001) and construction (r=0.838, p<0.001).

<table>
<thead>
<tr>
<th>SOLO score</th>
<th>N</th>
<th>M</th>
<th>SD</th>
<th>r</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program comprehension</td>
<td>57</td>
<td>3.4</td>
<td>0.58</td>
<td>0.799</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>Program construction</td>
<td>57</td>
<td>2.4</td>
<td>0.98</td>
<td>0.838</td>
<td>&lt;0.001</td>
</tr>
</tbody>
</table>

V. DISCUSSION

This research provided evidence that roles of variables can be used to improve the novices’ outcomes of b-learning on the C language programming. Data analysis showed that the improvement was contributed to novice programmers’ SOLO level of the code reading and code writing. This study demonstrated that the role-based teaching tends to experience significant achievement gains comparing to the classical b-learning teaching method. In the final exam, 55.5% of the students in the role-based group achieved equal or more than 80% accuracy, comparing to only 27.6% in the b-learning group. Meanwhile, 37.0% of the students in the role-based group achieved full SOLO score in the code reading, comparing to 16.7% in the b-learning group. Moreover, 22.2% of the students in the role-based group achieved full SOLO score in the code writing, comparing to 10.0% in the b-learning group.

However, no significant difference was found between the two groups in their understanding of the basic concepts of C language. This is due to the fact that students in the experimental group spent less time on studying the basic concepts. Identifying roles of variables and following program execution are not easy tasks for novices, so students need to devote more time to learn basics programming knowledge to learn about the roles of variables in the same course length. Acquiring and developing knowledge about programming is a highly complex. Even at the level of computer literacy, it requires construction of conceptual knowledge and the structuring of basic operations (such as loops, conditional statements, etc.) into schema and plans [24]. In this research, participants in the experimental group were asked to list all roles used in their homework of F2F learning and the programming construction in the web-based learning, in fact, was to ask participants to learning conceptual program knowledge by reasoning the roles of

<table>
<thead>
<tr>
<th>GROUP</th>
<th>SOLO LEVEL</th>
<th>PROGRAM COMPETENCE</th>
<th>PROGRAM CONSTRUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>M</td>
<td>SD</td>
<td>r</td>
</tr>
<tr>
<td>27</td>
<td>79.7</td>
<td>10.01</td>
<td>2.61</td>
</tr>
<tr>
<td>30</td>
<td>71.8</td>
<td>12.65</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>39.7</td>
<td>4.23</td>
<td>0.14</td>
</tr>
<tr>
<td>30</td>
<td>39.5</td>
<td>6.15</td>
<td></td>
</tr>
</tbody>
</table>
variables and program reading. Some studies reported that reading and explaining could significantly improve novice understanding abstract concept [25].

Programming is a complex cognitive process. In the classroom, it is necessary for a teacher to select the appropriated educational assessment model to evaluate student’s learning achievement and to conduct programming teaching. In this article, we found that students’ final exam score was positively correlated with the level of novice programmer’s ability to read a program and ability to write one. Similar results were also found by [12, 23]. These findings suggest that, the teacher can use the SOLO category to analyze students’ level of code reading and code writing and help students to develop personalized learning path, on the other hand, the teacher can also use SOLO in the final exam to evaluate the overall learning outcomes.

VI. CONCLUSION

B-learning provides a chance to improve novice program learning. Reading and explaining computer program with b-learning learning method, on the other hand, is a challenge for many novice programmers. According to previous researches, roles represent programming knowledge on a higher level than the simple programming language knowledge.

In this article, the SOLO taxonomy was used to evaluate effectiveness on b-learning environment on novice programmer with the concept of role of variable. The results show there are some significant improvements, including novice’s performances in the final examination, and ability in reading and writing a program. These results indicate that the b-learning with roles of variables to provide scaffolding is effective on novice program learning.

ACKNOWLEDGMENT

This research was financially supported by Scientific and Technological Research Projects of Henan Provincial Department of Education (2011A520031,14A10021, 16A520009,17A520006), Science and Technology Projects of Henan Provincial Department of Science and Technology (122102210136,142300410014,152107000098,1521022103 29,162102210047,162102310474), and Central Plains Economic Zone in Henan Province Tourism Wisdom Collaborative Innovation Center Open Project (2015-ZHLV-007, 2015-ZHLV-014).

REFERENCES


