公務出國或赴大陸地區報告提要

出國或赴大陸地區報告名稱：EdMedia 2014 (World Conference on Educational Multimedia, Hypermedia and Telecommunications) 含附件：■是□否

出國計畫主辦機關：臺北市立內湖高級中學
聯絡人：羅玗貞 電話：2797-7035轉699

出國人員姓名/服務機關/單位/職稱/電話
1. 羅玗貞/臺北市立內湖高級中學/秘書/2797-7035轉699
2. 林瓊甄/臺北市立內湖高級中學/資訊組長/2797-7035轉118
3. 侯莉莉/臺北市立內湖高級中學/教師/2797-7035轉515

出國類別：出席國際會議 □ 表演 □ 比賽 □ 競技 □ 洽展 □ 海外檢測 □ 三人共同撰寫論文發表 (公假自費)

出國期間：103/6/22至103/6/26
報告日期：103/7/29
出國地區：芬蘭

內容摘要：（300至500字）

2014 芬蘭資訊教學研討會 (EdMedia 2014 - World Conference on Educational Multimedia, Hypermedia and Telecommunications)，為國際資訊教育交流重要的研討會，此研討會由國際資訊教育促進協會 (Association for the Advancement of Computing Education, AACE) 主辦。AACE 成立於 1981 年，它是一個為資訊科技教育及 e 化學習社群服務，由來自世界各地 70 個國家，1500 個成員所組成的國際性組織。此國際研討會提供來自世界各地從事資訊科技研究、研究及實務教學的夥伴，關於運用資訊科技於教育場域討論、互動與心得交換的機會。本校教師合作撰寫之論文，榮獲入選，論文如附件一，於 103 年 6 月 22 日至 26 日前往芬蘭參與研討發表，與各國專業人士交換意見，除對教師專業成長及課程研發有所助益，亦能提升國際視野。

此次會議主要的議程有專題演講及論文報告，共有 399 篇論文在會議中發表。我們的論文發表時間為 103 年 6 月 26 日下午，論文報告以 100 年至 102 年內湖高中執行高瞻計畫的研究成果為論文內容，提出針對「基礎程式設計」及「進階程式設計」課程，結合新興雲端科技，研發創新「區塊式填寫」程式設計課程與教學，來改進教學現況，提升學生學習成果並減低程式初學者的認知負擔。研究結果顯示：利用雲端程式設計合作學習平台，學生更能體會撰寫程式設計的樂趣、提升寫程式的信心、增加對語法的熟悉度，及運用分析思考力解決問題。而平台上的針對性「區塊式填充」練習，可以讓學生專注於當下需要學習的語法與邏輯概念，並且增加撰寫程式的信心。

在參與研討會的過程中，除了吸收各地優秀學者所提供的研究資訊以及創新的想法之外，在這樣的過程中，激發更多不同思考與創意，我們期待可以將這些經驗轉化於我們設計之高瞻計畫教材及程式語言教學，以協助我們的學生，在資訊科技領域有更好的學習成果。
Reducing Novice Programmers’ Cognitive Load and Improving Learning Efficiency by Using Gap-Filled Programming Practice System

Yu-Jen Lo, Chiuang-Chen Lin, Lily Hou, Jane-Dong Wu
Taipei Municipal Nei-Hu High School
Taipei, Taiwan
bjj@nhs.tp.edu.tw

Yu-Chi Feng
National Experimental High School at Hsinchu Science Park
Hsinchu, Taiwan
casey_feng@hotmail.com

Te-Chin Chu, Greg C. Lee
Dept. of Computer Science & Information Engineering
National Taiwan Normal University
Taipei, Taiwan
leeg@csie.ntnu.edu.tw

Abstract: One of the main challenges for novice programmers is to learn the statement, syntax, logical thinking and program design in a short time. Students usually experience a heavy cognitive load when learning to program. The limitation of teaching hours in high school also leaves no time for instructors to individually assist each student. In this study, we propose a gap-filled programming practice (GFPP) system to help novice programmers reduce their cognitive load and ensure students focus on the specific contents instructors want them to practice. The research shows that students who were taught with GFPP system scored higher than students who were taught with traditional approach. The students can finish the practices efficiently and the positive feedbacks from GFPP system make them feel more confident and unafraid of learning programming.

Introduction

In the past, computer science was considered a university level subject. However, with the rapid growth of the applications of computer science and the increase role of computer technology in our daily life, many countries have introduced computer science at K-12 level. For example, the USA (Merritt et al., 1993) and Israel (Gal-Ezer, Beeri, Harel, & Yehudai, 1995) have developed computer science curriculum for high school since 1990s. In Austria (Micheuz, 2005), Informatics was introduced to all students in Grade 9 and became part of high school curriculum since 1985. When teaching computer science to high school students, computer programming plays a fundamental role. Computer programming is not only a production tool that enables students to implement the theories they learn from class, but also is helpful for students on the development of problem solving skills (Treese, 2003). Therefore, students who are not going to pursue computer science can still benefit from the process of learning programming.

Learning to write a complete program is never an easy task. Writing a program involves multiple cognitive activities, such as problem understanding, program design, implementing, testing and debugging. For novice programmers, a heavy cognitive load is experienced while constructing a program (Sweller, 1994). Most students only learned the concept of syntax after one semester of basic programming, and they do not know how to design a program (Smith, Cypher, & Tesler, 2001). In order to reduce students’ cognitive load while learning to program, Garner (2009) explored a part-complete solution method and developed a code restructuring tool (CORT) to help novice programmers. The CORT system permits the display of a partly completed code to a programming task, together with a set of possible lines of code that can be used to "fill-in" the
solution. This research suggests that the part-complete procedure proves to be useful as a way of scaffolding students’ learning of programming.

In addition, computer programming courses are characterized by a large amount of exercises that students are expected to intensively practice in order to develop good programming skills (Lam, Chan, Lee, & Yu, 2008). A traditional programming class usually consists of a lecture and in-class practices. The instructor explains the concept and students follow the instructions to do practices. In a 50-minute class, students usually can only finish one or two complete programs. There are two disadvantages of this teaching method. First, students only know the solution to the problem provided by the instructor. Students have no clue on how to solve a problem that they have never faced before. Second, the instructor cannot individually assist each student debugging their programming codes in class due to the time restriction. However, it is very important for students to get frequent and timely feedback on while writing their programs; if instructors can provide such feedback efficiently, students’ learning will be greatly improved (Bancroft & Roe, 2006).

Therefore, this research proposes a gap-filled programming practice (GFPP) system that can reduce student’s cognitive overload, increase practice opportunities and enhance the efficiency of learning programming language.

**Gap-filled method**

With gap-filled programming practice, instructors give the complete program structure and block out a part of the program to ask students to fill up the blank. A GFPP system was developed to supply a large number of exercises for students to practice with. Learning programming language in this way, students can focus on the blank, which is the part that instructors want them to focus on a particular lecture. For a novice programmer, the blank should be as simple as syntax or a variable declaration, instead of a whole programming structure. Instructors give a hint for each blocked-out part to reduce the chance of students’ mistakes. Instead of finishing a complete program, students enhance their comprehension by practicing the same concept repeatedly.

**Methodology**

This experiment chose a four-week lesson for array program. In the period of four weeks, students in experimental group intensively worked on targeted practice whereas control group practiced complete programs. Practice questions were the same for both groups except in different formats. Pretest and posttest were administered to investigate students learning achievement. Curriculum feedback was also conducted in the end of the experimental period to conclude students overall conception.

Two classes at high school in Taiwan were selected for the research. The number of participants was 73 tenth graders, splitting into two groups: 37 in experimental group and 36 in control group. Two instructors, whom were experienced high school instructors and have taught information technology subject for many years, co-teach the classes for this experiment.

All students were given a pretest before the experiment. The experimental group was taught using GFPP system. For every 50-minute class, instructors first lectured 10-15 minutes on C program language syntax concept; then students practiced gap-filled problems on the GFPP system. The gap-filled problems the experimental group practiced for every class included one line short answers and 2-5 lines segments. They could discuss with classmates via face-to-face approach, attached online chat room, or asked instructors at their convenience. As for the control group, students learn about programming in traditional class setting, instructors lectured first and students practiced writing complete program in Dev C++ development environment without using GFPP system. Students could discuss with classmates or consult with instructors. Both groups of students were practicing the same number of programming problems.
Results

The research data include a pretest, a posttest and a curriculum feedback questionnaire. The tests (pretest and posttest) contain questions about concepts of array structure, such as declaration of an array, data storage and retrieval from an array. The full mark of the tests was 50. An independent-samples t-test was conducted to compare the pretest scores for the experimental group and control group. There was no significant difference in scores for the experimental group (M=19.05, SD=9.16) and the control group (M=19.44, SD=9.05; t(71)=-0.18, p=0.79). In conclusion, before the experiment, the students’ programming ability in both groups showed no significant differences. After four-week learning, an independent-samples t-test analyses of posttest showed that the score of the experimental group (M=29.38, SD=15.54) was significantly higher than that of the control group (M=27.08, SD=12.32; t(71)=0.70, p=0.03). A paired-samples t-test was conducted to evaluate the impact of the teaching method on students' scores on the Fear of Statistics Test (FOST). For the experimental group, there was a statistically significant increase in FOST scores from pretest score (M=19.05, SD=9.16) to posttest score (M=29.38, SD=15.54, t(36)=-4.57, p<0.05). For the control group, there was also a statistically significant increase in FOST scores from pretest score (M=19.44, SD=9.05) to posttest score (M=27.08, SD=12.32, t(35)=-4.40, p<0.05). The assessment results showed that students who were taught with GFPP system scored higher than students who were taught with traditional approach.

At the end of the experiment, all students needed to fill out a curriculum feedback questionnaire. There were three parts in the questionnaire for the experimental group, which were: about the learning contents, about the practice questions and about the using of GFPP system. The questionnaire for the control group only has the first two parts. Table 1 showed the results of the questionnaire about the learning contents. Overall, 87% of the students in both groups were attentive in class (Q1). The students in the experimental group had a higher percentage of completing the practice questions (Q2: 95% versus 87% in the control group). The students in the control group had higher ratio (73% versus 64% in the experimental group) of understanding the learning contents (Q3). About 56% of the students in either group consider that arrays are difficult to learn (Q4). Most students in the control group (79% versus 59% in the experimental group) indicated that the practice questions on arrays are difficult (Q5).

Table 1. The statistical result of curriculum feedback questionnaire - learning contents

<table>
<thead>
<tr>
<th>Questions</th>
<th>Experimental group (N=37)</th>
<th>Control group (N=36)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Strongly agree (%)</td>
<td>Agree (%)</td>
</tr>
<tr>
<td>Q1. I was attentive in my class.</td>
<td>38 49 13 0 0</td>
<td></td>
</tr>
<tr>
<td>Q2. I practiced all the given questions.</td>
<td>36 59 5 0 0</td>
<td></td>
</tr>
<tr>
<td>Q3. I understood all content that was taught in class.</td>
<td>18 46 31 5 0</td>
<td></td>
</tr>
<tr>
<td>Q4. Arrays are difficult to learn.</td>
<td>10 46 41 3 0</td>
<td></td>
</tr>
<tr>
<td>Q5. The practices on arrays were</td>
<td>8 51 36 5 0</td>
<td></td>
</tr>
</tbody>
</table>
Table 2 showed the results of the questionnaire about the practice question. More students in the control group (73% versus 64% in the experimental group) agreed that the practice questions are various (Q1), while more students in the experimental group agreed that the practice questions are challenging (Q2: 77% versus 68% in the control group) and interesting (Q3: 57% versus 35% in the control group).

Table 2. The statistical result of curriculum feedback questionnaire - practice questions

<table>
<thead>
<tr>
<th>Questions</th>
<th>Experimental group (N=37)</th>
<th>Control group (N=36)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Strongly agree (%)</td>
<td>Agree (%)</td>
</tr>
<tr>
<td>Q1. The practicing questions of array program are various.</td>
<td>18</td>
<td>46</td>
</tr>
<tr>
<td>Q2. The practicing questions of array program are challenging.</td>
<td>33</td>
<td>44</td>
</tr>
<tr>
<td>Q3. The practicing questions of array program are interesting.</td>
<td>8</td>
<td>49</td>
</tr>
</tbody>
</table>

Only the experimental group students needed to give feedback of using GFPP system so that the instructors could understand the change of students’ learning process after using the GFPP system. Table 3 showed the results of the questionnaire about the using of GFPP system. Overall, more than 60% of the students thought that the use of GFPP system could make them unafraid of programming (Q1), increase self-confidence (Q2), help to grab logical concept (Q3), memorize syntax more easily (Q4) and arouse their interest in learning programming (Q5).
Table 3. The statistical result of curriculum feedback questionnaire - the review of using GFPP system

<table>
<thead>
<tr>
<th>Questions</th>
<th>Experimental group (N=37)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Strongly agree (%)</td>
</tr>
<tr>
<td>Q1. Compared with completing a program, filling up the blocked-out part of a program makes me unafraid of programming.</td>
<td>21</td>
</tr>
<tr>
<td>Q2. I think it can increase my self-confidence by using GFPP system.</td>
<td>21</td>
</tr>
<tr>
<td>Q3. I think it can help me grab logical concept by using GFPP system.</td>
<td>26</td>
</tr>
<tr>
<td>Q4. I think it helps me memorize syntax more easily by using GFPP system.</td>
<td>31</td>
</tr>
<tr>
<td>Q5. Generally, my interest in learning programming is aroused by GFPP system.</td>
<td>10</td>
</tr>
</tbody>
</table>

To sum up, although the students in both groups agreed that the learning contents of array are difficult, they were attentive in class and practiced most of the given questions. With regard to the practice questions, we observed that the students in the experimental group could pay attention on what the instructor wanted them to learn and thought the questions carefully. However, the students in the control group can’t focus on the content due to other parts of the program such as the input/output statement or spelling error. The higher percentage of completing the practice questions in the experimental group gives students positive feedbacks which make students feel the practice questions interesting and challenging.

Discussion
In order to obtain a deeper understanding on the advantages of using GFPP system, two instructors were interviewed and below are the compiled results.

In the traditional manners, students need to write the program from the first statement. It tends to make students focus on the wrong programming parts, like spelling errors, so that students do not have enough time to practice what they learned in this class. When learning on GFPP system, students can focus on the blocked-out part that the instructor wants them to learn instead of writing lots of codes that are not related to the learning content. The blocked-out part can be as simple as a variable declaration or as complex as a segment of program or a whole program. Therefore, depends on the intended difficulty of the exercise, instructors can block-out various parts of the program for students to work on.

In the traditional setting, if students have questions while writing programs, they can either wait for the instructor to check their program or ask other students for help. However, students usually have to wait for a long time if their peers don’t know how to solve the problem, and the instructor is too busy helping other students. With the gap-filled method, students can focus on the particular part of the program, which reduces the chance making irrelevant mistakes. In addition, when students practice in the off-line compiler environment, they believe that they have finished writing the program, but they do not know if their program is logically correct. This problem can
be solved by using GFPP system. After students submit their program, the system tests the correctness of the program automatically with several predefined dataset and gives students real-time feedbacks. When students submit the correct program, the system gives students a response like “congratulations!” Students said that when they see those positive feedbacks on the screen, they feel more confident.

In the traditional setting, students must install compiler in their own computer. With the GFPP system, students don’t have to install anything. Students can edit their program with any editor and submit to the GFPP system with any web browser.

Conclusions

The results of the data analyses reveal the impact of the gap-filled method on students’ learning. Experimental results show a significant better outcome on students’ learning achievement when learning with the GFPP system. Students can finish the practices efficiently and the positive feedbacks from GFPP system make them feel more confident and unafraid to learning computer programming.

References


