A Two-Tier Test-based Approach to Improving Students' Computer-Programming Skills in a Web-Based Learning Environment

Tzu-Chi Yang[1], Gwo-Jen Hwang[2], Stephen J. H. Yang[1*] and Gwo-Haur Hwang[3]
[1]Department of Computer Science and Information Engineering, National Central University, No. 300, Jung-Da Road, Chung-li, Taoyuan 320, Taiwan // [2]Graduate Institute of Digital Learning and Education, National Taiwan University of Science and Technology, 43, Sec.4, Keelung Rd., Taipei, 106, Taiwan // [3]Department of Information Networking and System Administration, Ling Tung University, No. 1, Ling-Tung Rd., Taichung, 400, Taiwan // tcyang.academic@gmail.com // gjhwang.academic@gmail.com // jhyang@csie.ncu.edu.tw // ghhwang@teamail.ltu.edu.tw
[*]Corresponding author

(Submitted August 15, 2013; Revised February 8, 2014; Accepted June 1, 2014)

ABSTRACT

Computer programming is an important skill for engineering and computer science students. However, teaching and learning programming concepts and skills has been recognized as a great challenge to both teachers and students. Therefore, the development of effective learning strategies and environments for programming courses has become an important issue. To address this issue, this study proposes a two-tier test-based learning system to enhance students' learning outcomes in computer-programming courses. We conducted an experiment on a college computer-programming course to evaluate the effectiveness of the proposed method. The experimental results show that the proposed method not only improves the students' attitude toward learning the programming language, but also enhances their programming skills.

Keywords

Computer programming, Computer-assisted learning, Two-tier test, Web-based learning environments

Introduction

The rapid development of information technology has created high demand for skillful programming specialists. Programming skills have therefore become a core competence for engineering and computer science students (Verdú et al., 2012; Hwang, Shadiev, Wang, & Huang, 2012; Fessakis, Gouli, & Mavroudi, 2013). However, learning a computer-programming language involves the comprehension of theoretical background and practice of a range of semantic and syntactic knowledge, coding skills, and algorithmic skills, which are usually complex and difficult for most students to master (Brooks, 1999; Govender, 2009; Katai & Toth, 2010; Wang, Li, Feng, Jiang, & Liu, 2012; Yeh, Chen, Hung, & Hwang, 2010). Researchers have reported that many instructors have encountered difficulties in teaching programming languages. Moreover, most students and teachers agree that learning programming is a challenging task that many students struggle with (Govender & Grayson, 2008; Kordaki, 2010). Therefore, it has become an important and challenging issue to develop effective strategies or tools for teaching computer-programming languages (Emurian, Holden, & Abarbanel, 2008; Hwang, Shadiev, Wang, & Huang, 2012).

Researchers have argued that when learning programming, continuous practice is required to ensure that the knowledge is retained (Chen, Chang, & Wang, 2008; Hwang, Wang, Hwang, Huang, & Huang, 2008). Moreover, actively and periodically scheduled learning is important for students to attain high levels of achievement (Hwang & Wang, 2004). Nevertheless, many computer science students cannot grasp the most fundamental concepts of programming and are thus unable to produce even the most basic programs (Eckerdal, 2009). Researchers have indicated that learning strategy, lack of study, and lack of practice are the causal attributes of success or failure in a computer programming course (Hawi, 2010; Hwang, Wu, Tseng, & Huang, 2011).

Among various learning strategies, providing accurate and meaningful prompts based on individual students' test results has been recognized by researchers as being an effective approach (Hung, Yang, Fang, Hwang, & Chen, 2014; Hwang, Sung, Hung, Yang, & Huang, 2013; Wu, Hwang, Milrad, Ke, & Huang, 2012). To this end, this study proposes a two-tier test approach with a feedback mechanism. An experiment has been conducted to evaluate the effectiveness of the proposed approach by investigating the following research questions:

- Do the students who learn with the two-tier test-based learning approach show better programming knowledge than those who learn with the conventional technology-enhanced learning approach?
- Do the students who learn with the two-tier test-based learning approach show better programming skills than those who learn with the conventional technology-enhanced learning approach?

Literature review

In the past decade, various teaching strategies and learning activities have been applied to computer-programming courses for beginners (Govender & Grayson, 2008). For example, Machanick (2007) proposed the idea of abstraction-first teaching by hiding details until students are ready for them. In addition, Emurian, Holden, and Abarbanel (2008) employed a peer-tutoring approach, and Hwang, Shadiev, Wang, and Huang (2012) proposed a web-based programming-assisted system to provide learning support for programming courses. In the meantime, researchers have indicated that problem-based learning (PBL) could be a promising approach for programming language learning (Kordaki, 2010; Pereira, Zebende, & More, 2010). For example, Gálvez, Guzmán, and Conejo (2009) reported a problem-solving environment to diagnose students' knowledge levels and to generate feedback and hints to help students understand and overcome their misconceptions in learning programming languages.

According to the literature, providing effective support or guidance by identifying the knowledge levels and learning difficulties of students in learning programming languages could be the key to the improvement in students' learning performance. Researchers have indicated the important role of assessment in identifying the learning status of individual students in providing effective learning guidance (Hwang, 2003; Hwang, Panjaburee, Triampo, & Shih, 2013; Tseng, Chu, Hwang, & Tsai, 2008). Researchers have further argued that assessment could be used as a learning strategy to improve students' learning performance (Chu, Hwang, Tsai, & Tseng, 2010; Gálvez, Guzmán, & Conejo, 2009). For example, Hauswirth and Adamoli (2013) proposed a pedagogical approach aligned with this aspect that allows students to learn from their mistakes by answering a series of questions. On the other hand, researchers have also indicated that assessing students' programming knowledge and skills as well as identifying their learning difficulties remains a challenging issue, implying the need to develop effective strategies or tools to identify students' status and provide learning support in programming language courses (Wang, Li, Feng, Jiang, & Liu, 2012).

Among the various approaches to programming training, tests are a popular activity adopted by teachers for examining students' learning status and guiding them to learn (Hwang & Chang, 2011; Gálvez, Guzmán, & Conejo, 2009; Trotman & Handley, 2008; Wang, Li, Feng, & Lui, 2012). Researchers have reported the effectiveness of tests in engaging students to learn and practice, by using multiple-choice items without detailed feedback (Roediger & Karpicke, 2006). The effectiveness of this approach could be attributed to its positive effects on students' learning retention when engaging them in thinking and practising (Butler, Karpicke, & Roediger, 2007). Gálvez, Guzmán, and Conejo (2009) further suggested that assessment itself can be used as a learning strategy.

Among the existing testing strategies, two-tier tests have been recognized as being an efficient and effective way of investigating students' prior knowledge or misconceptions, or alternative conceptions by many researchers, especially in science education (Chu & Chang, 2014; Odom & Barrow, 1995; Tsai, 2001). A two-tier test consists of a set of two-level multiple-choice questions. It was first introduced by Treagust (1988), mainly for diagnosing students' misconceptions or alternative conceptions in science (Tsai & Chou, 2002). The first tier assesses students' descriptive or factual knowledge of a phenomenon. The second tier probes the students' reasons for the choice they made in the first tier. The aim of a two-tier test is to explore students' in-depth explanations of the factual knowledge, which allows teachers or researchers to not only understand students' possible incorrect ideas, but also to assess the reasoning or in-depth understanding behind these ideas (Chu, Hwang, Tsai, & Tseng, 2010).

In the past decade, researchers have employed the two-tier test approach as an assessment or learning guidance tool. For example, Tsai and Chou (2002) developed a networked two-tier test system in which only one tier of a test item is presented per screen. Such a system facilitates assessment of the existing knowledge of a larger sample of students in a more efficient and relatively straightforward manner. Chou, Chan, and Wu (2007) employed a computerized two-tier test method to assess students' understanding and alternative conceptions of cyber copyright laws. Researchers have argued that a computerized or web-based two-tier test is not only feasible and efficient, but also provides an easy and familiar interface for students to answer the questions (Chou, Cha, & Wu, 2007). Moreover, because a two-tier test is in a multiple-choice format, it is much simpler for researchers or educators to interpret students' responses.

Regarding the cognition levels of programming skills, Brooks (1999) indicated that programming skills can be categorized into four levels (i.e., understanding of the task; finding methods; and coding, testing, and debugging the program). Govender and Grayson (2008) proposed five levels of learning programming (i.e., meeting the requirements, learning syntax, understanding and assimilating, problem solving, and programming in the large). Furthermore, several researchers (Machanick, 2007; Hwang, Wang, Hwang, Huang, & Huang, 2008) have assessed students' programming skills based on Bloom's taxonomy; that is, Knowledge, Comprehension, Application, Analysis, Synthesis and Evaluation. Each level of cognitive development is an important dimension for evaluating students' learning state of programming.

In this study, we used a two-tier test as a strategy for guiding students to learn a programming language. We also developed a web-based assessment and guidance system based on the proposed approach and conducted an experiment to evaluate the learning performance of the students in terms of their programming knowledge (Knowledge and Comprehension levels) and programming skills (Application levels) based on a knowledge test and a skills test.

Two-tier test-based programming language learning system

In this study, we propose a two-tier test-based programming language learning system to support web-based learning activities for computer-programming courses. We established a web-based learning environment with client-application-database server architecture. Moreover, we designed the user interface to adapt to the screen sizes of different devices; that is, students are able to interact with the system using personal computers, notebooks, tablets, or smartphones. Individual students can log into the system to review the teaching materials they have learned and to take tests. When students answer a test item, the system provides immediate corresponding feedback, including the correct answer, an explanation of the answer, and the supplementary material for the misconception if the student fails to correctly answer the item.

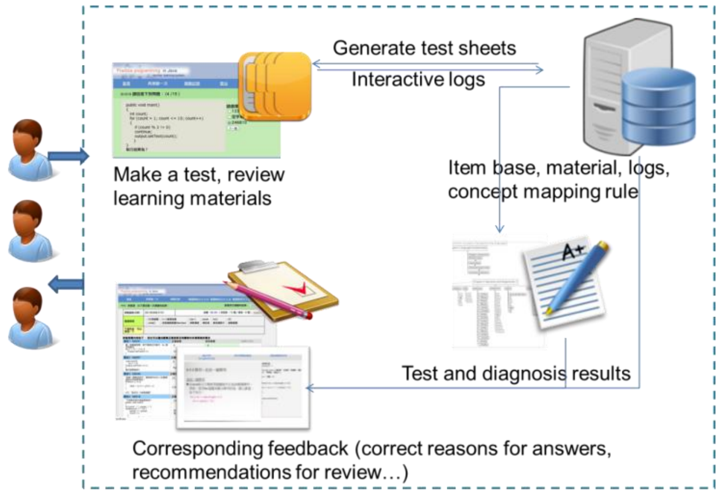

Generate test sheets
Interactive logs
Make a test, review learning materials
Item base, material, logs, concept mapping rule
Test and diagnosis results
Corresponding feedback (correct reasons for answers, recommendations for review…)

Figure 1. System structure of the online two-tier test for programming learning

System framework and functions

We developed a web-based two-tier test system for programming learning, OT3PL, which stands for online two-tier test for programming learning, to assist students in answering questions and justifying their answers with logical arguments. A total of 147 first-tier test items were developed in this study for assessing students' knowledge and skills of object-oriented programming. Among them, 125 items related to the knowledge level, while 22 items referred to the skills level. Moreover, 452 second-tier items were developed to further confirm the students' knowledge and ability or to identify their learning difficulties. The proposed system consists of three subsystems: the online test subsystem (OTS), the misconception diagnosis subsystem (MDS), and the learning content interactive subsystem (LCIS). Figure 1 shows the overall framework of the OT3PL.

Online test subsystem

After learning a unit or concept, students are asked to finish at least one test before they continue on to the next unit. The online test subsystem aims to generate two kinds of tests: (1) conventional multiple-choice questions selected from the item bank; and (2) a two-tier test, in which every test item has four choices in the first tier, followed by three or four reasons in the second tier for each choice. Table 1 shows an example of a two-tier test item for a programming language. In a conventional multiple-choice item, only the three choices in the first tier (i.e., (A) bar = blue, (B) bar = green, and (C) syntax error) are presented, while an additional nine choices are used to assess the reasons for the students' choice in a two-tier test item.

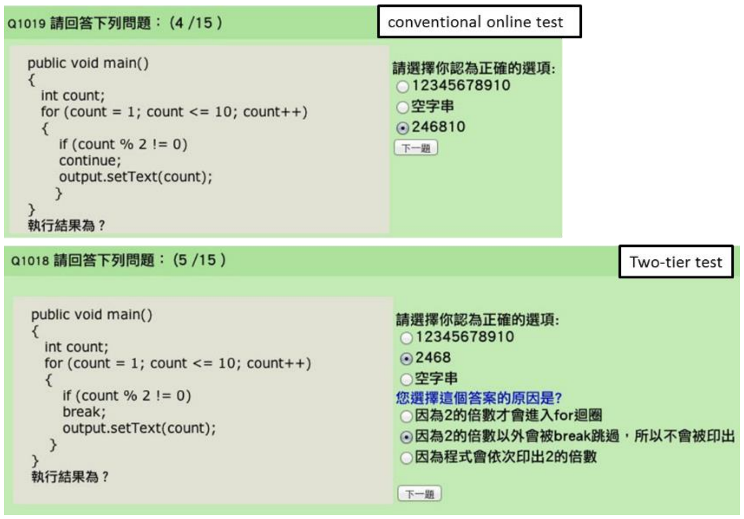
Table 1. Illustrative example of a two-tier test item

Given the following code:

```
    1. public class Test{
    2. public static void main(String[] args){
    3. String foo = “blue”;
    4. String bar = foo;
    5. foo = “green”;
    6. output.setText (“bar = “+bar);
    7. }
    8. }
```

What is the output?

First tier (the same in two kinds of test)		Second tier (asked in the two-tier test)
(A)	bar = blue	(A1) bar refers to string object “blue” via foo
		(A2) “blue” assigned to bar immediately
		(A3) bar assigned by a copy from foo
(B)	bar = green	(B1) bar assigned to the same memory address as foo
		(B2) bar and foo become the same object since the statement *String bar = foo;*
		(B3) bar refers to foo
(C)	Syntax error	(C1) The String should be string (in lower-case)
		(C2) Statement *“bar =” + bar* is incorrect,
		(C3) In line 5, *foo = “green”;* causes an error



Q1019 請回答下列問題：(4 /15)

```
public void main()
{
  int count;
  for (count = 1; count <= 10; count++)
  {
     if (count % 2 != 0)
     continue;
     output.setText(count);
    }
}
```

執行結果為？

請選擇你認為正確的選項:
- 12345678910
- 空字串
- [✓ 246810]

Q1018 請回答下列問題：(5 /15)

```
public void main()
{
  int count;
  for (count = 1; count <= 10; count++)
  {
     if (count % 2 != 0)
     break;
     output.setText(count);
    }
}
```

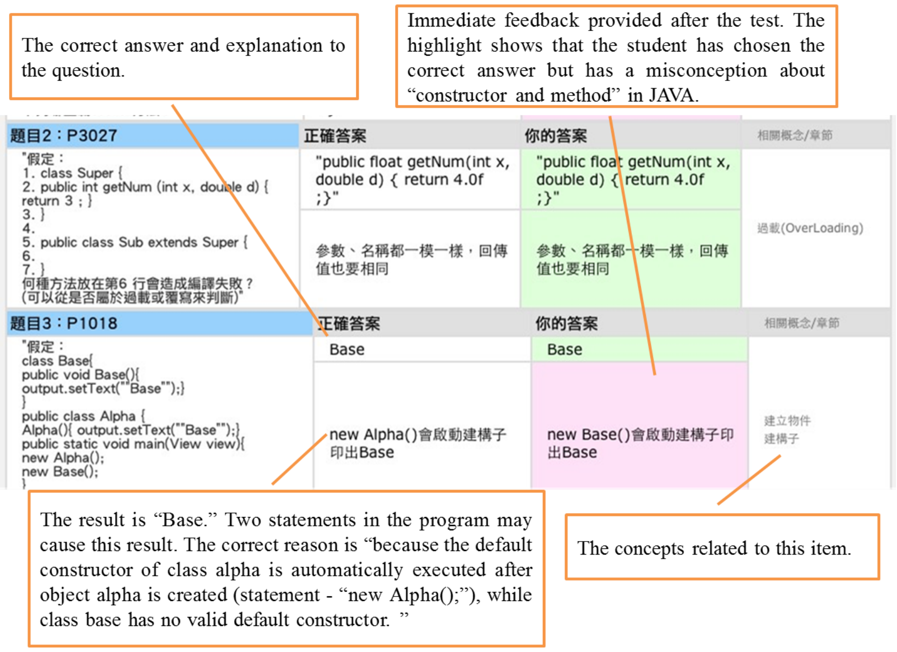
執行結果為？

請選擇你認為正確的選項:
- 12345678910
- [✓ 2468]
- 空字串

您選擇這個答案的原因是?
- 因為2的倍數才會進入for迴圈
- [✓ 因為2的倍數以外會被break跳過，所以不會被印出]
- 因為程式會依次印出2的倍數

Figure 2. Comparison of a conventional online test and a two-tier test

To prevent students from memorizing the answers, the sequence of the choices and the programming language variables in the test items are changed when generating test sheets. That is, no identical test sheets are generated during the learning process. When making choices in the second tier, students are allowed to view the corresponding first-tier item to identify the reason for making the choice. The scope of the tests is related to the weekly course progress. Students are able to extend the test scope by including more concepts or units that they have learned. Figure 2 shows the system interfaces of the conventional online test and the two-tier test. Figure 3 shows the test results and feedback from the learning system.



The correct answer and explanation to the question.

Immediate feedback provided after the test. The highlight shows that the student has chosen the correct answer but has a misconception about "constructor and method" in JAVA.

題目2：P3027	正確答案	你的答案	相關概念/章節

"假定：
1. class Super {
2. public int getNum (int x, double d) {
return 3 ; }
3. }
4.
5. public class Sub extends Super {
6.
7. }
何種方法放在第6 行會造成編譯失敗？
(可以從是否屬於過載或覆寫來判斷)"
	"public float getNum(int x, double d) { return 4.0f ;}"
	參數、名稱都一模一樣，回傳值也要相同
		"public float getNum(int x, double d) { return 4.0f ;}"
		參數、名稱都一模一樣，回傳值也要相同
			過載(OverLoading)

題目3：P1018	正確答案	你的答案	相關概念/章節

"假定：
class Base{
public void Base(){
output.setText(""Base"");}
}
public class Alpha {
Alpha(){ output.setText(""Base"");}
public static void main(View view){
new Alpha();
new Base();
}
	Base
	new Alpha()會啟動建構子印出Base
		Base
		new Base()會啟動建構子印出Base
			建立物件
			建構子

The result is "Base." Two statements in the program may cause this result. The correct reason is "because the default constructor of class alpha is automatically executed after object alpha is created (statement - "new Alpha();"), while class base has no valid default constructor. "

The concepts related to this item.

Figure 3. Illustrative example of a test result and feedback

Misconception diagnosis subsystem

In the misconception diagnosis subsystem (MDS), the first-tier item is used to assess the students' descriptive or factual knowledge of programming problems. The second tier probes the students' reasons for their choice made in the first tier. Therefore, the choice made in the second-tier item not only confirms whether the students have fully understood a concept, but also helps them to explore their misconceptions if incorrect reasons are selected. Therefore, a decision-tree rule base is used in MDS for diagnosing students' misconceptions or learning problems. The two-tier test items were developed by two experienced teachers by referring to the past test data in the course. Therefore, misconception identification in this study refers to finding the incorrect concepts, which leads students to make mistakes in programming.

The decision tree for judging the students' learning problems based on their answers is given in Figure 4. In this example, it is possible for two string objects to have the same value but to be located in different areas of memory in Java. The students may produce a logical error when developing software projects using variables that refer to other variables whose values are changed in later steps if they do not completely understand how to work with objects. For example, in a program, a variable <banana> = "green" and another variable <lemon> = <banana>, the value of <lemon> is therefore "green." Later, if <banana> is assigned as "yellow," the students might think that the value of <lemon> is "yellow," too. In fact, the value of <lemon> remains the same (i.e., "green"). Figure 3 showed that the choice of reason A2 implies that the student is not familiar with the process of string objects, especially value assignment, while if the student takes reason A3 into account, he/she may be confused about how the value of a string object is located in memory.

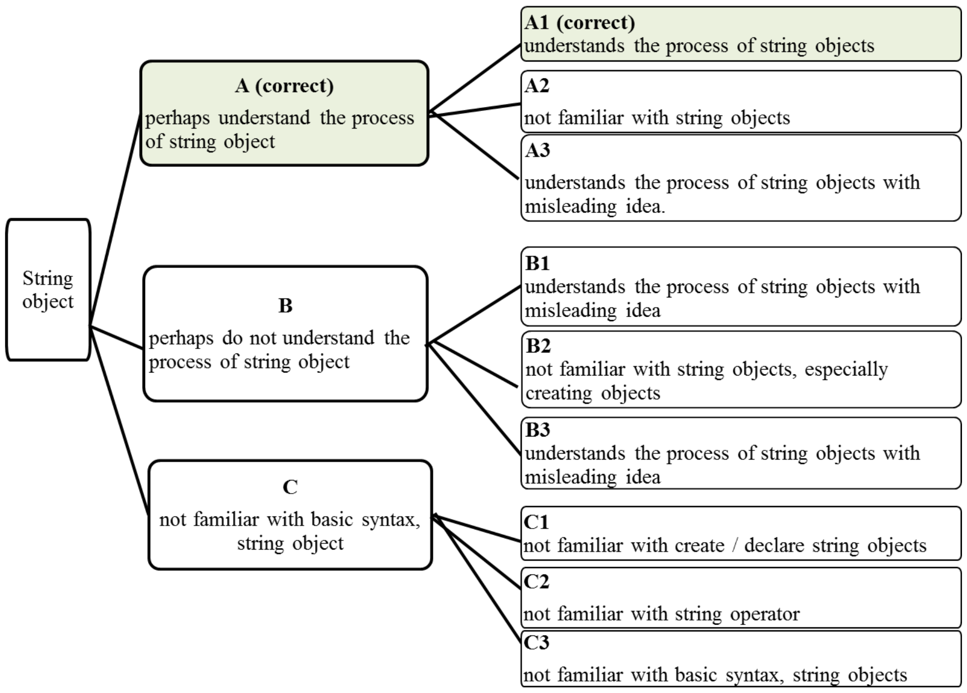

String object
A (correct)
perhaps understand the process of string object
B
perhaps do not understand the process of string object
C
not familiar with basic syntax, string object
A1 (correct)
understands the process of string objects
A2
not familiar with string objects
A3
understands the process of string objects with misleading idea.
B1
understands the process of string objects with misleading idea
B2
not familiar with string objects, especially creating objects
B3
understands the process of string objects with misleading idea
C1
not familiar with create / declare string objects
C2
not familiar with string operator
C3
not familiar with basic syntax, string objects

Figure 4. Decision tree for justifying the learning status of students

To develop the items for the two-tier test, eight graduate students with more than two years' programming experience were employed to construct the test items. For example, Table 2 shows four items revised by the teacher in order to identify whether the student completely understands the operator, flow control, and the scope of the variables. The four items in this example are very similar and seem easy to answer. However, even if students are familiar with the use of increasing operators, they will fail to construct flow control statements properly if they do not precisely understand the prefix and postfix of an operator.

Table 2. Sample test item

	Result
What is the value of i after executing the following code? int i; for(i=1;i<5;i=i+1);	i=5

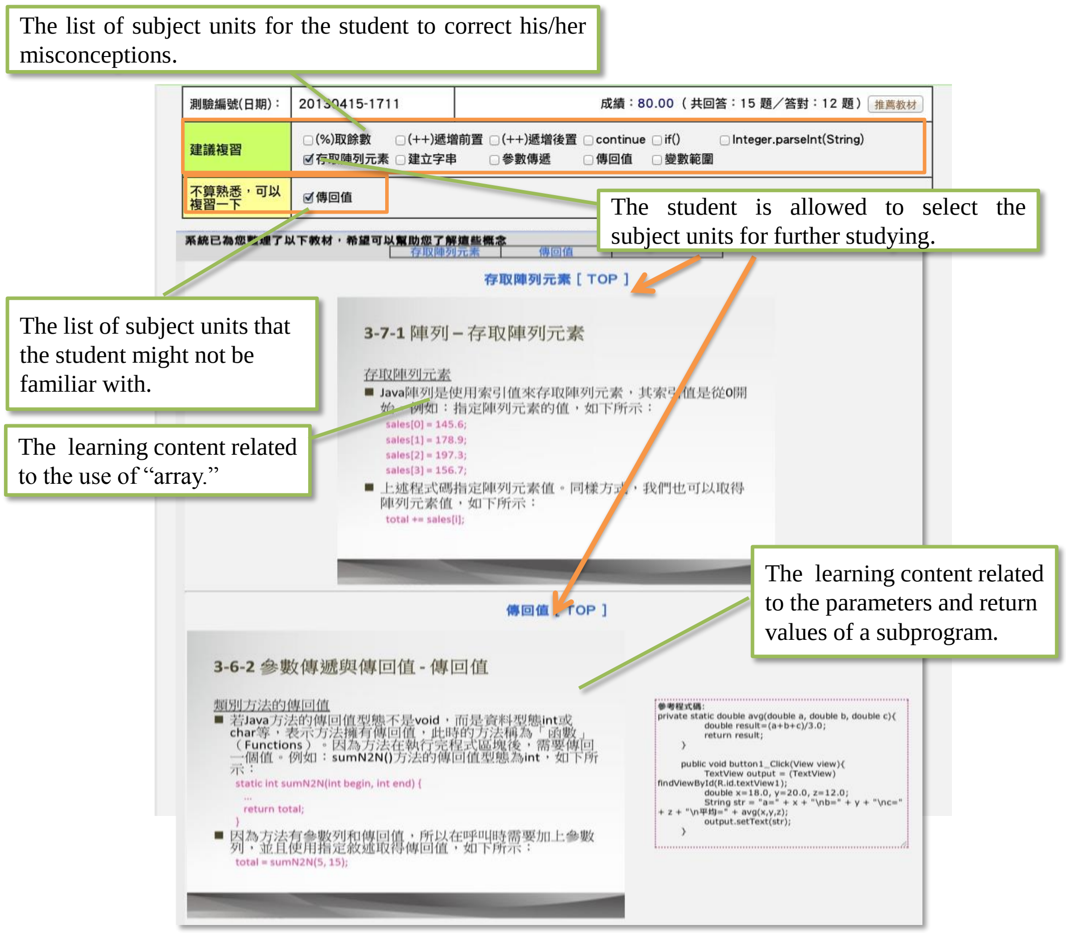
What is the value of i after executing the following code?
int i=4;
for(i=1;i<5;i+=1);
	i=5

What is the value of i after executing the following code?
int i;
for(i=1;i<5;i=++i);
	i=4

What is the value of i after executing the following code?
int i;
for(i=1;i<5;i=i++);
	unexpected looping

Learning content interactive subsystem

The learning content interactive subsystem (LCIS) presents up-to-date learning contents, including learning material extracted from the textbook and demonstrated programming code provided by the teacher. Students are able to review the correlated contents at any time, such as slides used in the class and the examples given by the teacher. Based on the students' misconceptions or learning status reported by MDS, LCIS provides relevant supplementary materials to help the students overcome their misconceptions. When students take a test, LCIS presents a list of highlights and provides students with the recommended learning materials. Figure 5 shows an illustrative example of providing personalized learning content to individual students based on test results.


The list of subject units for the student to correct his/her misconceptions.
The student is allowed to select the subject units for further studying.
The list of subject units that the student might not be familiar with.
The learning content related to the use of “array.”
The learning content related to the parameters and return values of a subprogram.

測驗編號(日期)：	20130415-1711	成績：80.00（共回答：15 題／答對：12 題）推薦教材
建議複習	☐ (%)取餘數 ☐ (++)遞增前置 ☐ (++)遞增後置 ☐ continue ☐ if() ☐ Integer.parseInt(String) ☑ 存取陣列元素 ☐ 建立字串 ☐ 參數傳遞 ☐ 傳回值 ☐ 變數範圍	
不算熟悉，可以複習一下	☑ 傳回值	

系統已為您整理了以下教材，希望可以幫助您了解這些概念

存取陣列元素 | 傳回值

存取陣列元素 [TOP]

3-7-1 陣列 – 存取陣列元素

存取陣列元素

- Java陣列是使用索引值來存取陣列元素，其索引值是從0開始。例如：指定陣列元素的值，如下所示：

```
sales[0] = 145.6;
sales[1] = 178.9;
sales[2] = 197.3;
sales[3] = 156.7;
```

- 上述程式碼指定陣列元素值。同樣方式，我們也可以取得陣列元素值，如下所示：

```
total += sales[i];
```

傳回值 [TOP]

3-6-2 參數傳遞與傳回值 - 傳回值

類別方法的傳回值

- 若Java方法的傳回值型態不是void，而是資料型態int或char等，表示方法擁有傳回值，此時的方法稱為「函數」（Functions）。因為方法在執行完程式區塊後，需要傳回一個值。例如：sumN2N()方法的傳回值型態為int，如下所示：

```
static int sumN2N(int begin, int end) {
  ...
  return total;
}
```

- 因為方法有參數列和傳回值，所以在呼叫時需要加上參數列，並且使用指定敘述取得傳回值，如下所示：

```
total = sumN2N(5, 15);
```

參考程式碼:

```
private static double avg(double a, double b, double c){
            double result=(a+b+c)/3.0;
            return result;
      }

      public void button1_Click(View view){
            TextView output = (TextView)
findViewById(R.id.textView1);
            double x=18.0, y=20.0, z=12.0;
            String str = "a=" + x + "\nb=" + y + "\nc="
+ z + "\n平均=" + avg(x,y,z);
            output.setText(str);
      }
```

Figure 5. Example of providing personalized comments and learning materials

Experiment design

To evaluate the efficacy of the proposed approach, we conducted an experiment to compare the learning achievements and attitudes of the students who were learning the programming language using the proposed approach. We conducted the experiment on the learning activities of the “Developing Android applications using Java” course of a college in central Taiwan. Details of the experiment are stated below.

Participants

Eighty-eight college students participated in the online programming learning activity. The average age of the students was 20. They were randomly divided into an experimental group and a control group. Eventually, 79 students completed all of the activities, with 40 in the experimental group and 39 in the control group. All of the students were taught by the same teacher.

Treatment

The aim of the course was to train students to develop application programs for mobile devices. The target programming language was Java. The learning objectives included understanding the Java syntax, learning debugging skills, and programming with Java statements to complete specified problems.

Figure 6 presents the experiment design of this study. Both groups of students first received instruction on the basic knowledge of the programming language (Java), then took a pre-test and completed a questionnaire to analyze their knowledge of and attitudes towards the programming language before interacting with the proposed system.

In the first stage of the activity, both groups of students received face-to-face classroom instruction. The teacher provided instruction on the basic syntax and executive environment of Java. The students were then asked to complete a segment of code and perform an exercise based on what they had learned in the classroom.

In the second stage, both groups of students were asked to take a test on the system after each class. In general, they completed at least five exercises and one test each week. In this stage, the students in the experimental group learned with OT3PL. On the other hand, the students in the control group learned with the conventional technology-enhanced learning approach; that is, they were allowed to browse the same learning materials and do the same exercises, but they receives conventional tests and feedback in the web-based learning system.

The learning activity was conducted for four weeks for both groups. After the learning activity, both groups of students took a post-test. They were then asked to fill in the same questionnaire to measure if their attitudes towards programming learning had changed.

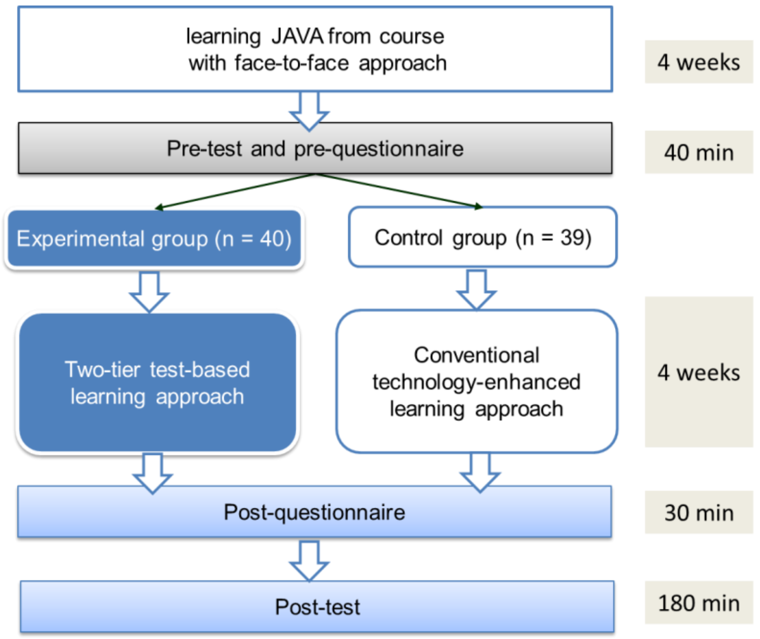

learning JAVA from course with face-to-face approach
4 weeks
Pre-test and pre-questionnaire
40 min
Experimental group (n = 40)
Control group (n = 39)
Two-tier test-based learning approach
Conventional technology-enhanced learning approach
4 weeks
Post-questionnaire
30 min
Post-test
180 min

Figure 6. Experiment procedure

Measuring tools

Both the pre-test and the post-test were developed by consulting two teachers who had taught the programming language course for more than five years. The pre-test consisted of 33 multiple-choice items, with a total score of 100. The pre-test was designed to evaluate the students' basic knowledge concerning computer science and programming before the learning activity. The post-test, a detailed achievement test on the Java language, consisted of a programming knowledge test and a programming skills test. The programming knowledge test consisted of 30 multiple-choice items for conceptions, data structure, and coding of Java. The programming skills test consisted of five programming problems. The perfect scores for the two dimensions were 30 and 70, respectively.

The questionnaire for "attitudes toward the programming language course" consisted of seven items. A six-point Likert scheme was used to rate the questionnaire items, where 6 meant strong agreement or positive feedback and 1 represented high disagreement or negative feedback. The Cronbach's alpha value of the questionnaire was 0.94.

Experimental results

The students' learning performance was evaluated based on two dimensions, that is, the programming knowledge and programming skills. The former was concerned with the students' knowledge of programming, such as variables, syntax, and programming concepts. Namely, the programming knowledge dimension focused on the ability of remembering and comprehension of the Java language, and was evaluated by a set of multiple choice items as shown in Table 3.
On the other hand, programming skills refer to the ability of completing software programs based on the specified purposes or function. In other words, to complete a programming task, students require not only programming knowledge and the syntax of the target programming language, but also the ability of identifying problems, organizing codes, and solving problems (Hauswirth & Adamoli, 2013). In this study, the students' programming skills were evaluated based on the correctness of the programs they developed. For example, they were asked to complete several Android application programs, such as "Please design project T3, such that the screen presents the user's name in blue when the user touches the screen, but presents the user's name in red when the user touches the screen for more than two seconds" and "Given the following four pictures, please design project T4, such that users can switch pictures by sliding their finger." Based on Bloom's taxonomy, programming knowledge involves recognizing and remembering the usage of Java statements, which develops the Knowledge level and the Comprehension level. Moreover, the programming skills required include solving problems and organizing code statements, relating mainly to the Application level (Anderson & Krathwohl, 2001; Bloom, 1994).

Table 3. Examples of multiple choice items for evaluating programming knowledge

Item 10. What is the result of calling the following codes?
int i=1,j=10;
do{
    if (i>j) {
    break;
}
      j--;
}while(++i<5);
Output.setText("i="+i+"and j="+j);
	A. i = 6 and j = 5
	B. i = 5 and j = 5
	C. i = 6 and j = 4
	D. i = 5 and j = 6

Item 21. What is the result of calling the following codes? String s = "hello" +9 +1 ; Output.setText (s) ;	A. Hello91 B. Hello10 C. Compiling error D. Throws an exception

Item28. public class MethodOver {
public void setVar（int a , int b , float c）{ }
}
Which of the following is an overloading method of serVar？
	A. private void setVar (int a , float c , int b) {}
	B. protected void setVar (int a , int b , float c) {}
	C. public int setVar (int a , int b , float c) {return a;}
	D. protected void setVar (int a , int b , float c) { return c;}

Before participating in the learning activity, the students took a pre-test to evaluate their basic knowledge of computer science and programming languages. The means and standard deviations of the pre-test scores were 67.80 and 13.03 for the experimental group, and 66.87 and 11.49 for the control group. A *t*-test performed on the pre-test scores showed no significant difference between the pre-test results of the two groups, with $t = 0.335$ and $p > .05$; that is, the two groups of students had equivalent knowledge of the computer language prior to the learning activity.

After conducting the two-tier test learning activity, the two groups' programming knowledge test and programming skills test scores were compared. It was found that the students in the experimental group had significantly better programming skills than the control group, while no significant difference was found between their programming knowledge test scores, as shown in Table 4. According to the aforementioned, the programming knowledge represents the abilities of remembering and comprehending the Java language, while programming skills refers to the ability of organizing programming codes and solving problems. The two-tier test system aimed to help students identify their misconceptions of programming, which could be related to the syntax and semantics of the programming language, or the ability of organizing the programming codes for dealing with a practical problem.

From the experimental results, we found that the two-tier test system benefited the students more in terms of improving their ability of solving problems by organizing the codes in correct ways. In addition, we observed during the learning activity that one third of the students in the experimental group completed all of the tasks in the programming skills test, while most of the students in the control group completed only three tasks.

It should be noted that, in this study, the two-tier test approach was a learning guidance method rather than a test method. The purpose of using the approach was to help the students identify their learning difficulties or problems such that they were able to efficiently cope with the problems via reading supplementary materials or discussing with peers. The effectiveness of such learning guidance or prompts has been reported by several previous studies in science or mathematics courses (Chu et al., 2010; Panjaburee et al., 2013). The interview results from 11 participants provide further evidence for the effectiveness of the two-tier test approach in helping them comprehend the programming knowledge and improve their programming skills. For example, several students have argued that the learning system improved their programming knowledge and skills.

S01: "There are many statements that I did not really understand, because I do not usually use them. However, I have understood the statements since taking the test and following the guidance provided by the learning system."
S02: "I always study very hard. However, I did not find my weakness in programming and the misconceptions of the programming language until I learned with this system and did the practice."
S03: "The system refined my knowledge and improved my ability in the programming areas that I did not learn well."
S08: "It is not easy to learn a programming language by only reading the materials in books and practising without immediate feedback. It is important to do such practice. The test and feedback process is really helpful to me in improving my programming ability."

Other students also shared similar thoughts that the learning approach not only helped them identify their learning problems, but also improved their programming ability by organizing what they had learned to solve the problems and complete the projects. From the feedback of the students and the experimental results, it is concluded that the two-tier test approach benefited the students by correcting their misconceptions of programming and guiding them to effectively improve their programming knowledge and skills during the learning process.

Such a finding implies that the OT3PL approach can improve the students' programming skills by enhancing their logical reasoning and by helping them make connections between the programming language (e.g., syntax and types), programming concepts (e.g., variables, data structure, and control flow) and programming procedures (e.g., solving programming problems).

Table 4. The *t*-test results of the programming knowledge and programming skills tests for the two groups

	Group	*N*	Mean	*SD*	*t*
Programming knowledge test	Experimental group	40	20.49	2.95	−1.442
	Control group	39	21.48	3.12	
Programming skills test	Experimental group	40	55.05	21.00	2.546*
	Control group	39	44.49	15.36	

*p < .05.

Table 5 shows the *t*-test result of the students' attitudes toward learning the programming language before and after the learning activity. We found that the learning attitude of the students in the experimental group significantly improved after the learning activity, while the change in the control group students' attitudes was not significant. This finding conforms to previous studies concerning technology-enhanced learning that effective learning guidance strategies or mechanisms are helpful to students in terms of improving their learning attitudes as well as their learning achievements (Chu, Hwang, Tsai & Tseng, 2010; Hwang & Chang, 2011).

Table 5. The paired *t*-test result of the students' attitudes toward programming learning before and after the learning activity

		Experimental group (*N* = 40)			Control group (*N* = 39)		
		Mean (*SD*)	*t*	*p*	Mean (*SD*)	*t*	*p*
Attitudes toward programming learning	Before	4.68 (1.26)	−3.36**	0.003	4.90 (1.29)	0.54	0.59
	After	5.33 (1.06)			4.82 (1.11)		

**p < .01.

Conclusions

In this paper, we proposed a two-tier test approach for programming learning. Moreover, an online two-tier test system for programming learning with misconception diagnostics and a feedback mechanism, OT3PL, has been implemented based on the proposed approach. OT3PL assists students in assessing their misconceptions of knowledge of programming and in improving their programming skills by judging numbers of statements of code and verifying their reasons. We conducted an experiment on a course on developing Android applications to evaluate the effectiveness of the proposed approach by comparing the learning performance of the students who learned with OT3PL and those who learned with the conventional technology-enhanced learning approach. From the experimental results, we found that, with the help of OT3PL, the experimental group students showed significantly better programming skills than those in the control group.

The finding implies that the two-tier test approach can benefit the students in terms of improving their procedure knowledge rather than their declarative knowledge. Procedural knowledge is the type of knowledge related to how to perform a task, solve problems, or complete a project (Chu, Hwang, Huang, & Wu, 2008; Lewicki, Hill, & Bizot, 1988). In this study, the programming problems in the programming skills test engaged the students in organizing the aims and functions of the problem and applying various programming concepts, syntax, and flow control skills to the development of the program; therefore, the test is relevant to procedural knowledge as well as to the application level of cognition in Bloom's taxonomy. This finding is reasonable since the two-tier test approach can help students find the learning problems through the interactive questioning and feedback-providing mechanism. On the other hand, the programming knowledge test is related to declarative knowledge, which represents the description of facts or information and is related to the Knowledge and Comprehension levels of cognition. It is also reasonable to find that the two groups of students did not show significant difference in terms of programming knowledge since both approaches (two-tier test-based learning and conventional online test learning) provided the same learning materials to the students to memorize and comprehend.

Although OT3PL benefited the students in this application, there are some limitations to be noted. First, generalization of the findings may be limited to populations of a similar nature, but may not be very applicable for other learner groups with different educational settings or cultural backgrounds. Secondly, the number of students participating is rather small. Third, to provide feedback and judgment, the teachers need to spend time reviewing and refining the test items for evaluation purposes, and the digital learning materials for providing learning supports.

Further research is therefore needed in this area, particularly concerning students of diverse academic degrees and knowledge background. Moreover, it is worth extending this study to the Analysis and Synthesis levels by engaging students in analyzing and improving some existing programs, for example, by reorganizing and reducing code statements for improving the efficiency of the programs. On the other hand, it is also interesting to provide other learning supports, such as collaboration and annotation mechanisms to the learners (Sung & Hwang, 2013; Yang, Yu, & Sun, 2013). In the near future, we will apply this approach to other programming courses to investigate in depth how students look at the system and how they think the system helps them in our future work. Moreover, we plan to investigate the patterns of online behaviors and identify which patterns of behavior possibly result in different learning performance in such a learning activity.

Acknowledgements

This study was supported in part by the National Science Council of the Republic of China under contract numbers NSC 101-2511-S-008-011-MY3, and NSC 101-2511-S-011-005-MY3.

References

Anderson, L. W., & Krathwohl, D. R. (Eds.) (2001). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives.* New York, NY: Addison Wesley Longman.

Bloom, B. S. (1994). Reflections on development and use of the taxonomy. In L. W. Anderson & L. A. Sosniak (Eds.), *Bloom's taxonomy: A forty-year retrospective* (pp. 1–8). Chicago, IL: The National Society for the Study of Education.

Brooks, R. (1999). Towards a theory of the cognitive processes in computer programming. *International Journal of Human Computer Studies, 51*, 197–211.

Butler, A. C., Karpicke, J. D., & Roediger, H. L. (2007). The effect of type and timing of feedback on learning from multiple-choice tests. *Journal of Experimental Psychology: Applied, 13*(4), 273–281.

Chen, G. D., Chang, C. K., & Wang, C. Y. (2008). Using adaptive e-news to improve undergraduate programming courses with hybrid format. *Computers & Education, 51*, 239–251.

Chou, C., Chan, P. S., & Wu, H. C. (2007). Using a two-tier test to assess students' understanding and alternative conceptions of cyber copyright laws. *British Journal of Educational Technology, 38*(6), 1072–1084.

Chu, H. C., & Chang S. C. (2014). Developing an educational computer game for migratory bird identification based on a two-tier test approach. *Educational Technology Research & Development, 62*(2), 147–161.

Chu, H. C., Hwang, G. J., Huang, S. X., & Wu, T. T. (2008). A knowledge engineering approach to developing e-libraries for mobile learning. *The Electronic Library, 26*(3), 303–317.

Chu, H. C., Hwang, G. J., Tsai, C. C., & Tseng, J. C. R. (2010). A two-tier test approach to developing location-aware mobile learning systems for natural science courses. *Computers & Education, 55*, 1618–1627.

Eckerdal, A. (2009). *Novice programming students' learning of concepts and practise* (Doctoral dissertation), Retrieved from http://uu.diva-portal.org/smash/record.jsf?pid=diva2:173221

Emurian, H. H., Holden, H. K., & Abarbanel, R. A. (2008). Managing programmed instruction and collaborative peer tutoring in the classroom: Applications in teaching Java^{TM}. *Computers in Human Behavior, 24*, 576–614.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63*, 87–97.

Gálvez, J., Guzmán, E., & Conejo, R. (2009). A blended e-learning experience in a course of object oriented programming fundamentals. *Knowledge-Based Systems, 22*, 279–28.

Govender, I., & Grayson, D. J. (2008). Pre-service and in-service teachers' experiences of learning to program in an object-oriented language. *Computers & Education, 51*, 874–885.

Govender, I. (2009). The learning context: Influence on learning to program. *Computers & Education, 53*, 1218–1230.

Hauswirth, M, & Adamoli, A. (2013). Teaching Java programming with the Informa clicker system. *Science of Computer Programming, 78*, 499–520.

Hawi, H. (2010). Causal attributions of success and failure made by undergraduate students in an introductory-level computer programming course. *Computers & Education, 54*, 1127–1136.

Hung, I. C., Yang, X. J., Fang, W. C., Hwang, G. J., & Chen, N. S. (2014). A context-aware video prompt approach to improving in-field reflection levels of students. *Computers & Education, 70*(1), 80–91.

Hwang, G. J. (2003). A concept map model for developing intelligent tutoring systems. *Computers & Education, 40*(3), 217–235.

Hwang, G. J., & Chang, H. F. (2011). A formative assessment-based mobile learning approach to improving the learning attitudes and achievements of students. *Computers & Education, 56*(4), 1023–1031.

Hwang, G. J., Panjaburee, P., Triampo, W., & Shih, B. Y. (2013). A group decision approach to developing concept-effect models for diagnosing student learning problems in mathematics. *British Journal of Educational Technology, 44*(3), 453–468.

Hwang, G. J., Sung, H. Y., Hung, C. M., Yang, L. H., & Huang, I. (2013). A knowledge engineering approach to developing educational computer games for improving students' differentiating knowledge. *British Journal of Educational Technology, 44*(2), 183–196.

Hwang, G. J., Wu, C. H., Tseng, Judy C. R., & Huang, I. W. (2011). Development of a ubiquitous learning platform based on a real-time help-seeking mechanism. *British Journal of Educational Technology, 42*(6), 992–1002.

Hwang, W. Y., & Wang, C. Y. (2004). A study on learning time pattern in asynchronous learning environments. *Journal of Computer Assisted Learning, 20* (4), 292–304.

Hwang, W. Y., Shadiev, S., Wang, C. Y., & Huang, Z. H. (2012). A pilot study of cooperative programming learning behavior and its relationship with students' learning performance. *Computers & Education, 58*, 1267–1281.

Hwang, W.Y., Wang, C. Y., Hwang, G. J., Huang, Y. M., & Huang S. (2008). A web-based programming learning environment to support cognitive development. *Interacting with Computers, 20*, 524–534.

Katai, T, & Toth, L. (2010). Technologically and artistically enhanced multi-sensory computer-programming education. *Teaching and Teacher Education, 26,* 244–251.

Kordaki, M. (2010). A drawing and multi-representational computer environment for beginner learning of programming using C: Design and pilot formative evaluation. *Computers & Education, 54,* 69–87.

Lewicki, P., Hill, T, & Bizot, E. (1988). Acquisition of procedural knowledge about a pattern of stimuli that cannot be articulated. *Cognitive Psychology, 20*(1), 24–37.

Machanick, M. (2007). Teaching Java backwards. *Computers & Education, 48*, 396–408.

Odom, A. L., & Barrow, L. H. (1995). The development and application of a two-tiered diagnostic test measuring college biology students' understanding of diffusion and osmosis following a course of instruction. *Journal of Research in Science Teaching, 32*(1), 45–61.

Panjaburee, P., Triampo, W., Hwang, G. J., Chuedoung, M., & Triampo, D. (2013). Development of a diagnostic and remedial learning system based on an enhanced concept effect model. *Innovations in Education and Teaching International, 50*(1), 72–84.

Pereira, H. B. B., Zebende, G. F., & More, M. A. (2010). Learning computer programming: Implementing a fractal in a Turing machine. *Computers & Education, 55*, 767–776.

Roediger, H. L., & Karpicke, J. D. (2006). Test-enhanced learning: Taking memory tests improves long-term retention. *Psychological Science, 17*(3), 249–255.

Sung, H. Y., & Hwang, G. J. (2013). A collaborative game-based learning approach to improving students' learning performance in science courses. *Computers & Education, 63*(1), 43–51.

Treagust, D. F. (1988). Development and use of diagnostic tests to evaluate students' misconceptions in science. *International Journal of Science Education, 10*(2), 159–169.

Trotman, A., & Handley, C. (2008). Programming contest strategy. *Computers & Education, 50*(3), 821–837.

Tsai, C. C. (2001). The interpretation construction design model for teaching science and its applications to Internet-based instruction in Taiwan. *International Journal of Educational Development, 21*, 401–415.

Tsai, C. C., & Chou, C. (2002). Diagnosing students' alternative conceptions in science. *Journal of Computer Assisted Learning, 18*, 157–165.

Tseng, Judy C. R., Chu, H. C., Hwang, G. J., & Tsai, C. C. (2008). Development of an adaptive learning system with two sources of personalization information. *Computers & Education, 51*(2), 776–786.

Verdú, E., Regueras. L. M., Verdú, M. J., Leal, P. J., Castro, J. P., & Queirós, R. (2012). A distributed system for learning programming online. *Computers & Education, 58,* 1–10.

Wang, Y., Li, H., Feng, Y., Jiang, Y., & Liu, Y. (2012). Assessment of programming language learning based on peer code review model: Implementation and experience report. *Computers & Education, 59*, 412–422.

Wu, P. H., Hwang, G. J., Milrad, M., Ke, H. R., & Huang, Y. M. (2012). An innovative concept map approach for improving students' learning performance with an instant feedback mechanism. *British Journal of Educational Technology, 43*(2), 217–232.

Yang, X. M., Yu, S. Q., & Sun, Z. (2013). The effect of collaborative annotation on Chinese reading level in primary school of China. *British Journal of Educational Technology, 44*(1), 95–111.

Yeh, Y. F., Chen, M. C., Hung, P. H., & Hwang, G. J. (2010). Optimal self-explanation prompt design in dynamic multi-representational learning environments. *Computers & Education, 54*(4), 1089–1100.