Empirical Study of Educational Programming Language for K12: Between Dolittle and Visual Basic

SeungWook Yoo†, Kyoung-A Kim†, Yong Kim†, YongChul Yeum†, Susumu Kanemune††, and WonGyu Lee†††

†Department of Computer Science Education, Graduate School, Korea University
††Computer Center, Hitotsubashi University
†††Department of Computer Science Education, College of Education, Korea University

Summary
Many novice programmers avoid and sometimes give up on programming learning because he/she feel many cognitive burdens in programming learning. To relieve this burden and to enhance the ability to tackle problems, we are trying to search the suitable educational programming language for K12, especially to high school students. Language that is selected for study is Dolittle and VB. Experimental lesson was gone by teaching method that cognitive load theory is applied. Cognitive load is tested by the concept of programming, algorithm implementation and program management. As an experimental lesson, we selected 2nd grade high school female students who have similar learning achievements in pre-test. The students were divided into two groups. One group had 10 Dolittle lessons and the other group had 20 Visual Basic lessons. After an experimental lesson, it was concluded that Dolittle is more effective than Visual Basic in understanding programming concept and program management. This result is same in post-test that is evaluated in programming lesson after 1 year.

Key words:
Computer Science Education, Programming Education, Cognitive Load

Introduction
Effective method to teach principle of computer science is to teach by experience through programming progress. But, because of difficulty of programming and learner’s cognitive development step, there are not many countries that handle programming as independence subject. In addition, general purpose language such as VB or C language was included in some part of curriculum. This becomes reason that learner hard to study programming effectively [6][7][8].

Visual Basic and C language have been chosen for the national curriculum textbook in the field of high school programming language in Korea. However, many novice programmers are likely to give up learning the concept of programming and application even before learning programming design and implementation. Visual features and complicated functions of programming language are a disservice to novice programmers and do more harm than good. But Dolittle [3] as EPL(Educational Programming Language) helps novice programmers to learn programming in a shorter amount of time and lessens the cognitive burdens because of the simple application and writing source code based on text.

We used Dolittle in order to teach an important program concept, algorithm thinking, and implementation in programming education. By now Visual Basic was regarded as visual programming language, but it is also used to solve the program problem in the Korean textbook. It is difficult for novice programmers to use object-oriented programming since the program output does not appear to be created easily. But by using Dolittle, novice programmers are able to solve challenging problems themselves and they are motivated to learn. Furthermore, novice programmers understand the principles of running a computer and learn high grade programming. The light cognitive load of Dolittle helps novice programmers learn it easily and enhance the programming performance [2]. This empirical study is compared VB with Dolittle to find the suitable language for K12. The experimental lesson was designed by the cognitive load theory. Also the same learning goal was compared by VB and Dolittle.

2. Related Works

2.1. Dolittle for EPL

Dolittle’s design policy is the suitable simple syntax and the easy use method for primary school and secondary school. Therefore, learner must feel cognitive load as less as possible. But also it includes various functions to prepare actuality program. Dolittle was developed by Ganemune in 2000, and it is not dependent to operating system because is written in Java [3][4]. The main design policy of Dolittle is listed as follows:
Simple syntax with multi-language
 Incremental Programming
 Text-based Programming
 Objected-Oriented
 Open expandability

More detailed explanation is first, it uses multi-language (for example, Korean, Japanese, and English) to lower learner’s cognitive load. Second, in programming learning, K12 feels class, function, and variable definition hardly specially. To solve this K12 does program code write by one line. Then, make them confirm the result immediately. Third, Dolittle is educational programming language.

Learner must learn to method that express program exactly and compactly. For this it is operated in text-based programming environment. Forth, it used the prototype-based object-orientation. Learner transmits message to turtle object that appear on screen, and the object becomes manipulate. Dolittle makes new object through clone of object, and learner gets inheritance concept through relation between two objects. Fifth, communication with peripheral device or network should be available. Through this, students do cyber world experience at program of screen inside. Figure 1 is a sample program. From now on, we use a sample program and explain the syntax of Dolittle simply.

```
turtle_1 = turtle ! create.
star = [turtle_1 ! 200 forward 144 rightturn] ! 5 repeat.
star_obj = star ! makefigure (blue) paint.
```

Dolittle programs call on objects using “!”. Then this sample program sends a “create” message to the prototype object “turtle” to let it clone itself. Also assigns the created turtle object to the variable called “turtle_1.” The “!” indicates the end of the statement. This one command line draw a turtle object on screen. The turtle object is an object which implements turtle graphics.

```
turtle_1 = turtle ! create.
star = [turtle_1 ! 200 forward 144 rightturn] ! 5 repeat.
star_obj = star ! makefigure (blue) paint.
```

```
clock_1 = timer ! create 1 period 10 duration.
run_b = button ! "RUN" create.
run_b : click = [clock_1 ! [star_obj ! 36 rightturn] execute].
```

At this statement, the block is assigned to the “click” attribute of the “run_b” object. Dolittle defines a method by assigning a block to an attribute. The “click” is the method. When the button is clicked, the “execute” message is sent to the “clock_1”, and it repeats the execution of the block. As a result, the star rotate 36 degrees at a time.

```
turtle_1 = turtle ! create.
star = [turtle_1 ! 200 forward 144 rightturn] ! 5 repeat.
star_obj = star ! makefigure (blue) paint.
clock_1 = timer ! create 1 period 10 duration.
run_b = button ! "RUN" create.
run_b : click = [clock_1 ! [star_obj ! 36 rightturn] execute].
```

Figure 1. Sample Program by Dolittle

```
turtle_1 = turtle ! create.
star = [turtle_1 ! 200 forward 144 rightturn] ! 5 repeat.
star_obj = star ! makefigure (blue) paint.
```

```
clock_1 = timer ! create 1 period 10 duration.
run_b = button ! "RUN" create.
run_b : click = [clock_1 ! [star_obj ! 36 rightturn] execute].
```

A “timer” is an object which executes code pieces at a specified interval for a specified period.

```
run_b = button ! "RUN" create.
```

A “button” is an object which displays a GUI part of a button shape with label “RUN”.

```
run_b : click = [clock_1 ! [star_obj ! 36 rightturn] execute].
```

At this statement, the block is assigned to the “click” attribute of the “run_b” object. Dolittle defines a method by assigning a block to an attribute. The “click” is the method. When the button is clicked, the “execute” message is sent to the “clock_1”, and it repeats the execution of the block. As a result, the star rotate 36 degrees at a time.

```
turtle_1 = turtle ! create.
star = [turtle_1 ! 200 forward 144 rightturn] ! 5 repeat.
star_obj = star ! makefigure (blue) paint.
clock_1 = timer ! create 1 period 10 duration.
run_b = button ! "RUN" create.
run_b : click = [clock_1 ! [star_obj ! 36 rightturn] execute].
```

Figure 2. Result of Sample Program by Dolittle

```
star_obj = star ! makefigure (blue) paint.
clock_1 = timer ! create 1 period 10 duration.
run_b = button ! "RUN" create.
run_b : click = [clock_1 ! [star_obj ! 36 rightturn] execute].
```

When “makefigure” is sent to “star”, it separates the drawn lines from itself and returns the new figure object. The created figure object is received “paint” message. When the identifiers is refer within an argument list, it need to enclose with brackets("(...)”). Finally, this statement assigns the created figure object to the variable “star_obj”. The execution of this step paints the star blue.

```
turtle_1 = turtle ! create.
star = [turtle_1 ! 200 forward 144 rightturn] ! 5 repeat.
star_obj = star ! makefigure (blue) paint.
clock_1 = timer ! create 1 period 10 duration.
run_b = button ! "RUN" create.
run_b : click = [clock_1 ! [star_obj ! 36 rightturn] execute].
```

Dolittle has the control structure, “repetition”, “condition”, and “diverse” that is defined in the method. According to a previous study, Dolittle is regarded as the suitable programming language for the novice programmer to learn computing conception by the statement of structured algorithm [4][5].
2.2. Cognitive Load

The cognitive load renders it difficult for a novice programmer to perform programming because of the inherent complexity and strict accuracy of the program language. Garner divides the cognitive load into three subcategories: the intrinsic cognitive load, which a programmer feels about the programming language itself and the task to be performed; the extraneous cognitive load, which lies in the question of how to express the task; and the germane cognitive load concerning the resolving process [1].

Cognitive overload is regarded as the main reason for difficulty in programming learning and abandonment. It may dissolve the difficulty of language itself. For this reason, we have chosen programming language “Dolittle”.

3. Test Design and Implementation of Experimental Lesson

We designed the experimental lesson in following order. First, teaching-learning design was made to theory about cognitive load. Next, experimental group and control group were selected through pre-test.

This test selected the 2nd graders from 4 classes at the Information and Industry High School in Korea and divided them into two groups. Upon pre-test, the means of the groups showed similarities.

The experimental lesson was conducted with the second-year text, “General Course on Computer”. Two groups’ student studies same contents and solves same task in the textbook through Dolittle and Visual Basic.

<table>
<thead>
<tr>
<th>Dependent variable</th>
<th>Group</th>
<th>Mean</th>
<th>Standard Deviation</th>
<th>Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>Achievement of Learning</td>
<td>experimental group</td>
<td>57.87</td>
<td>21.64</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>control group</td>
<td>58.12</td>
<td>19.55</td>
<td>30</td>
</tr>
</tbody>
</table>

We conducted the class over 10 hours for Dolittle, and 20 hours for Visual Basic.
May of 2005 and lasted four weeks. To reduce the extraneous cognitive load, we provided the worked program samples for each assignment. On the other hand, to increase the germane cognitive load, part-complete programs were proposed. Figure 2 illustrates the overall study design.

Figure 4. Design of Experimental Lesson

The measurement tools for this study were two paper exams (25 questions) which evaluated background knowledge about computers and developed questions about evaluating the ability of creating and reading a program. The evaluation of learning achievement is based on a developed evaluation standard so that Dolittle and Visual Basic are evaluated objectively.

Table 3. Evaluation Standard for Making a Program

<table>
<thead>
<tr>
<th>Field</th>
<th>Evaluation list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program concept</td>
<td>Object use, create and grant property, understand of absolute coordinates and operation, distinguish objects</td>
</tr>
<tr>
<td>Algorithm and implementa-tion</td>
<td>Can calculate the total marks and average?</td>
</tr>
<tr>
<td></td>
<td>- sequential processing handling</td>
</tr>
<tr>
<td></td>
<td>- use operators</td>
</tr>
<tr>
<td></td>
<td>- operate field(reading, writing)</td>
</tr>
<tr>
<td></td>
<td>Can get the result using conditional statement?</td>
</tr>
<tr>
<td></td>
<td>- handling condition</td>
</tr>
<tr>
<td></td>
<td>- operate field(initialization)</td>
</tr>
<tr>
<td></td>
<td>- understand of block</td>
</tr>
<tr>
<td></td>
<td>As delete button event write about method (do know field object’s method?)</td>
</tr>
<tr>
<td></td>
<td>- operate field</td>
</tr>
<tr>
<td></td>
<td>- operate button</td>
</tr>
<tr>
<td></td>
<td>- understand of the initializing target</td>
</tr>
<tr>
<td>Manage program</td>
<td>The degree of file complement</td>
</tr>
</tbody>
</table>
gave up. Both learners are similar in the field of implementing algorithm & program. As a result, Dolittle learners are more successful than Visual Basic learners and are less prone to give up learning.

When evaluating the ability of reading a program, Dolittle learners are also more successful because learners are allowed to program in their native language.

We obtained the significant results with a questionnaire like the one below.

- The learners felt language challenges before they began to learn programming, and both learners who learn Dolittle and VB still felt challenges. However, 42% of the VB learners felt challenges, and 19% of Dolittle learners felt challenges.

- The degree of success in making a program is 90% in Dolittle and 58% in VB, which tells us that Dolittle users are more effective learners.

- 65% of Dolittle learners said that programming helps to understand the concept of the computer, but only 35% of VB learners said that.

- 3% of Dolittle learners have challenges in the field of procedure condition and repetition structure, while 29% of VB learners have challenges in the same field.

- 3% of Dolittle learners have challenges in the field of grammar, thinking logically, managing programs, and debugging that makes learning programming difficult, while 39% of VB learners have challenges in the same field.

- 35% of Dolittle learners are positive and 3% are negative in self evaluation when they were asked if they developed logical thinking or problem solving skills after learning programming, but 26% of VB learners were positive and 26% were very negative.

5. Further Study

Students who took part in experimental lesson carried out the new programming subject 1 year later. Making the business program is target of the lesson to use VB. A Post-test was done. Its purpose is that measure the student’s achievement of learning

A post-test was conducted in May, 2006. The teacher in charge did not know about the original test results. Students were asked to prepare working programs with Visual Basic. The post-test was conducted over a period of three months, and Table 5 shows the results.

<table>
<thead>
<tr>
<th>Dependent variable</th>
<th>Group</th>
<th>Mean</th>
<th>Standard Deviation</th>
<th>Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>Achievement of Learning</td>
<td>experimental group</td>
<td>61.58</td>
<td>21.64</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>control group</td>
<td>55.52</td>
<td>24.37</td>
<td>30</td>
</tr>
</tbody>
</table>

Our findings reveal that Dolittle forms a strong schema in the long-term memory of a learner, leading to better performance. The standard deviation indicates that it correlates with the fact that a fewer number of the subject students in the middle group gave up, which is expected to be explored in a future study.
References


SeungWook Yoo received the B.S. degree in Mechanical Engineering Education from Chungnam National University in 1983 and the M.S. degree in Computer Science Education from Korea University in 2002. He has been studying in the Department of Computer Science Education of Korea University. Main research field is the educational programming language in K12 especially.

Kyoung-A Kim graduated from HongIrk University in 1992, majoring in Education of Mathematics/Computer Science with B.S. degree. She started to study at the Graduate School of Education, Korea University, majoring in computer science education in March 2004 and will graduate in August 209 with master degree.

Yong Kim has been studying in the Department of Computer Science Education of KOREA university. His research fields are Informatics curriculum, Gifted education, Programming Language and e-Learning Quality Assurance. He has been working for KERIS (Korea Education & Research Information Service, an organization specialized ICT in education under the Ministry of Education) as a senior researcher since 1997.

YongChul Yeum received the B.S. degree in Mathematics Education from Seoul National University of Education in 1991 and the M.S. degree in Computer Science Education from Seoul National University of Education in 2002. He has been studying in the Department of Computer Science Education of Korea University. Main research field is the educational programming language in K12 especially.

Susumu Kanemune received the PhD in Systems Management from University of Tsukuba in 2004. His research interests are in the programming language and information education. He is an Associate Professor of Hitotsubashi University.

WonGyu Lee received the B.A. degree in English Language and Literature from Korea University in 1985, M.S. and Ph. D. degree in System and Information Engineering from University of Tsukuba in 1993 respectively. During 1993-1995, he stayed in the Korean Culture & Art Foundation. Since 1996, he is a professor of Computer Science Education at Korea University.