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:

Manuscript received June 5, 2006.
Manuscript revised June 25, 2006.

- 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.

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.

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.

star = [turtle_1 ! 200 forward 144 rightturn] ! 5 repeat.

A block is presented by the square brackets ("[….]"). "[….] ! 5 repeat" means that sends a message "Execute yourself five times" to the block. In this sample statement, Dolittle interprets identifiers after "!" (such as "forward", and "rightturn") as message selectors. And numeric literals or parenthesized expressions (such as "200", and "144") become arguments to the message. When an object processes a message, it normally returns the object itself. At this statement, the result of "forward"(the object itself) is sent to "rightturn". At this step, star shape's line is created and this is assigned to "star".

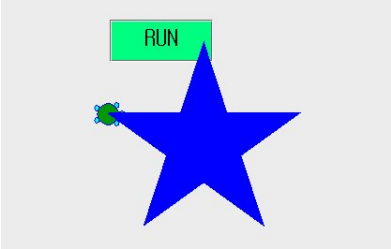

Figure 2. Result of Sample Program by Dolittle

star_obj = star ! makefigure (blue) paint.

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.

clock_1 = timer ! create 1 period 10 duration.

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 the "clock_1" repeats the execution of the block. As a result, the star_obj ratates 36 degrees at a time.

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 sub-categories: 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].

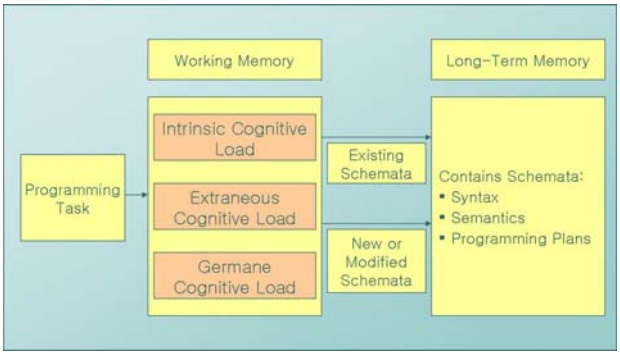

Figure 3. Programming Learning Process

The intrinsic cognitive load occurs due to the lack of the schema on the grammatical structure and the programming method of a program language, generating various errors such as syntax or semantic errors. Studies have reported on development and application of an educational language with the reduced intrinsic cognitive load [2].

The extraneous cognitive load refers to the burden concerning expression of a problem to be resolved through the programming process. It poses a heavy burden to someone without programming experience. For a novice programmer extraneous cognitive load causes an overload not in the long-term memory, but in the working memory in the course of resolving the extraneous cognitive load with the intrinsic cognitive load, which is already heavily applied. Worked examples serve as a useful tool for tackling the problem.

The germane cognitive load occurs while tackling tasks. In an educational program where the intrinsic load is not overwhelming or stimulating, the cognitive process of a learner can form a long-term schema during the programming process by lowering the extraneous load. As a result, the germane cognitive load is generated. It is necessary to enhance the schema in the long-term memory through the part complete solutions.

A learner without a schema in long-term memory may face cognitive difficulties in language and in expressing ideas. EPL may relieve these problems [4][5].

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.

Table 1. Result of Pre-Test

Dependent variable	Group	Mean	Standard Deviation	Students
Achievement of Learning	experimental group	57.87	21.64	30
	control group	58.12	19.55	30

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 2. Learning Contents

Should be to learn contents	Time(50 min/ unit)	
	Dolittle	VB
Learning to manage a program, initial picture, terms	1	1-3
Knowing to create objects, events, concept of method and grant properties	2	4-6
Understanding GUI tools and concepts of operators	3	7-8
Solving the problem using sequential process	4	9-10
Solving the problem using conditional statement	5	11-12
Solving the various problem to know conditional statement	6	13-14
Solving the problem using repetition statement	7	15-16
Practice programming	8	17-18
Evaluating the ability of making a program	9	19
Evaluating the ability of reading a program	10	20

We conducted the class over 10 hours for Dolittle, and 20 hours for Visual Basic. The experimental lesson began in

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.

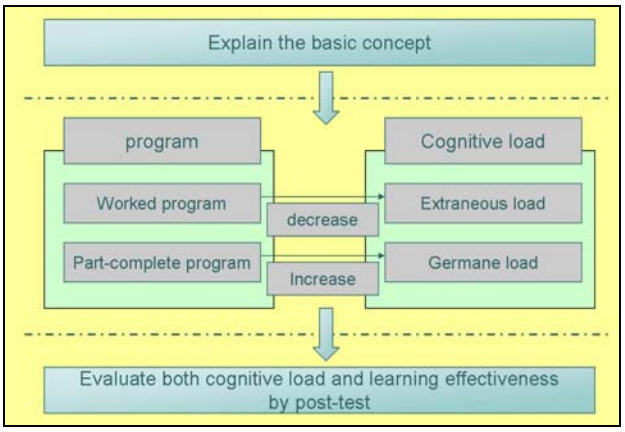

Figure 4. Design of Experimental Lesson

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

Field	Evaluation list
Program concept	Object use, create and grant property, understand of absolute coordinates and operation, distinguish objects
Algorithm and implementa-tion	Can calculate the total marks and average? - sequential processing handling - use operators - operate field(reading, writing) Can get the result using conditional statement? - handling condition - operate field(initialization) - understand of block As delete button event write about method (do know field object's method?) - operate field - operate button - understand of the initializing target
Manage program	The degree of file complement

Table 4. Evaluation Standard for Reading a Program

field	Evaluation list
Program concept	Creating object and grant property
Algorithm and implementa-tion	Repetition statement with condition Knowing condition exactly to solve problems? As delete button, event, write about method (do know field object's method?)
Manage program	none

Also, a questionnaire was conducted to know the opinions of programming learners.

4. Findings and Conclusion

The result of the experimental lesson is that Dolittle learners were more effective in learning than Visual Basic learners. Learners were evaluated by the making and reading of a program. The evaluation lists were the concept of the program, the implementation of algorithm, and managing the program. When evaluating the ability of making a program, Dolittle and Visual Basic have similar results for high grade students, but are very different for low grade students when it comes to the concept of program-creating object, grant property, and distinguishing objects.

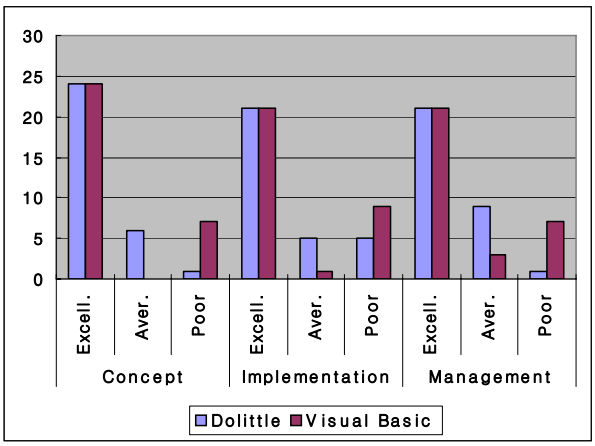

Figure 5. Result of Making a Program

While 23% of the Visual Basic learners gave up the concept of creating a program, only 3% of Dolittle learners gave up. We checked the existence of files and the degree of success in making a program when evaluating managing programs, and only 3% of Dolittle learners gave up while 23% of the Visual Basic learners

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.

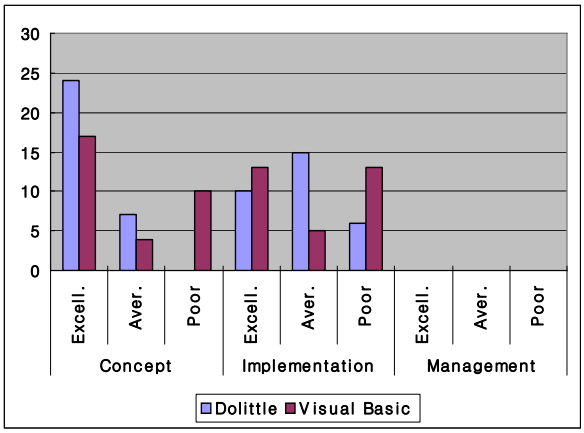

Figure 6. Result of Reading a Program

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 Dolitte 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.

To conclude, learning Dolittle has more positive effects on understanding a computer's concepts and increases problem solving skills and logical thinking, rather than VB, which is in the national curriculum.

Figure 7 shows one example of program that students should be complete. It is a program that input the name and score of three subjects and calculates the total score, mean, rank. Left of picture is completed by VB, and right is Dolittle.

Figure 7. Student’s Work for Making a Program

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 5. Result of Post-Test after 1 year

Dependent variable	Group	Mean	Standard Deviation	Students
Achievement of Learning	experimental group	61.58	21.64	30
	control group	55.52	24.37	30

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

[1] S. Garner, *Cognitive Load Reduction in Problem Solving Domains*. ICCE2001 Seoul, South Korea, 2001.

[2] S.W. Yoo, K.A. Kim, Y.C. Yeum, Y. Kim, and W. G. Lee., *Comparison of Cognitive Load Between Dolittle and Visual Basic*. Proceedings of The International Workshop on Information Science Education & Programming Languages, Korea University & University of Tsukuba, 2006.

[3] Dolittle programming language. http://dolittle.eplang.jp/

[4] S. Kanemune, T. Nakatani, R. Mitarai, S. Fukui, Y. Kuno, *Dolittle-Experiences in Teaching Programming at K12 Schools*, The Second International Conference on Creating, Connecting and Collaborating through Computing, IEEE, 2004.

[5] D.Y. Kwon, H. M Gil, Y. C. Yeum, S. W. Yoo, S. Kanemune, Y. Kuno, W. G. Lee, *Application and Evaluation of Object-Oriented Educational Programming Language Dolittle for Computer Science Education in Secondary Education*, JKACE 7(6), 2004

[6] http://education.nic.in/ncert_it_curri_guide/it_curri_content.asp

[7] England National curriculum, http://www.nc.uk.net/

[8] http://english.moe.go.kr/html/education/?menuno=03

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 HongIk 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.