International Journal of Engineering Innovation & Research
Volume 4, Issue 1, ISSN: 2277 – 5668

Programming Prospects of Learners Derived from Reference History to Estimate Design Ability

Yoko Itado
email:itado@de.is.ritsumei.ac.jp

Phuong Dinh Dong
email:dinhdongphuong@yahoo.com

Yusuke Kajiwara
email:kajiwara@de.is.ritsumei.ac.jp

Hiromitsu Shimakawa
email:simakawa@cs.ritsumei.ac.jp

Abstract: **Learners need design abilities for programming. It is necessary for them to have prospects indicating what programming knowledge they organize into a code. Since programming knowledge to be used in each assignment has been decided in advance, careful consideration is necessary to make learners to achieve design abilities through assignments. If the knowledge is specified in assignments, programming exercise courses cannot train learners to achieve design abilities. Meanwhile, if the assignments do not specify it, learners without programming prospects may be at loss what they should begin with. Supervisors should discriminate those learners from others in programming exercise courses.**

This paper proposes a method to determine whether learners work on assignments along with programming prospects. The method acquires reference histories to programming knowledge learners consult to get programming prospects. A classifier trained with a machine learning algorithm discriminates reference histories with programming prospects from others.

We verify the usefulness of the method, comparing the result using the method with what is manually derived from source codes. From this result, reference history for a specific time before this time can find programming prospect of learners. Supervisor cannot only grasp programming prospects of learners, but also provide suitable advices for individual learners, monitoring the transition of programming prospect.

Keywords: **Configuration, Design Ability, Education, Exercises Site, Programming Prospect, Reference History, Students.**

I. Introduction

The recent society demands more software engineers good at programming to enjoy enlarging convenience brought by Information technologies. Many educational institutions such as universities introduce programming education courses to respond the demand. Programming requests acquiring knowledge on programming languages and platforms on which programs runs. In addition to that, it needs an ability to organize the knowledge into a source code. Since various pieces of knowledge are used to write a program, a prospect to organize them is indispensable. The programming courses should teach learners so that they understand knowledge necessary for programming, but also write a source code with a prospect.

Knowledge necessary for programming is provided for learners from one teaching unit to another in lecture classes. In exercise classes, learners engage in assignments to understand how to use the knowledge they studied in the teaching unit. However, in this teaching style, we cannot expect learners can acquire the ability to organize the knowledge into a code, because the assignments fully specify the knowledge to be used in them. It would be dangerous for learners to work on assignments specifying no knowledge to be used, to get ability to organize the knowledge into code. Such assignments have high possibility to make learners at a loss to determine which knowledge they use. Deep depression might make them drop out from programming. Assignments specifying no knowledge to be used should be presented only to learners who have already understood the knowledge. In addition to that, we should not neglect to examine whether the learners are at a loss to figure out what they should do.

This paper proposes a method to nurse learners, examining whether the learners organize their knowledge on programming with prospects. The method focuses on learner reference to teaching materials explaining knowledge to be used in programming Learners with prospects would make pinpointed accesses to materials explaining exact knowledge, while learners without prospect would make haphazard accesses to materials. The method classifies learners from the viewpoints prospect for programming, using feature of reference histories of the learners. Furthermore, it supports supervisors to give appropriate guidance to individual learners, monitoring the transition of learner prospects for programming. Through successful guidance to individual learners, the method can assist learners to acquire the ability to organize programming knowledge into a code.

The remaining of the paper is organized as follows. Section 2 describes the present state and problems of programming exercise. The method is presented in section 3. Section 4 illustrates experiments to verify the method. The paper discusses the experiment result in section 5. Section 6 gives the conclusions and future work.

II. Design Ability in Programming Exercise

A. Present State of Programming Exercise

For learners without any programming experience to study programming, they have to acquire both of programming knowledge and skills to make a program using the programming knowledge. Introductory programming course for the learners consists of lecture classes and exercise classes.

Learners learn the programming knowledge in lecture classes. The programming knowledge is divided into teaching units. Learners study each teaching unit in a lecture class every week. First, they learn basic programmi ng knowledge. Assuming the understanding it, the lecture classes provide learners new programming knowledge founding on the basic one.

An exercise classes provides assignments to be solved using the programming knowledge learned in a lectures class. In an exercises class, learners try to embody the specification of an assignment as a program, making the best use of programming knowledge learned in a lecture class. Learners must use programming knowledge learned in the week as well as that learned in previous weeks to make a program. To make a program, learners have to make a plan what programming knowledge they use, and how they combine them. Namely, in an exercise class, they must organize various kinds of programming knowledge into a program. The paper refers to the ability to organize necessary programming knowledge into a program as the design ability.

Excellent learners can grasp each of the programming knowledge. They can make plans to organize the programming knowledge before they start writing codes to solve assignments. On the other hand, poor learners cannot organize the programming knowledge for a program, even if the assignment exemplifies programming knowledge necessary to solve it and a way to organize the programming knowledge. It is very difficult for them to organize programming knowledge into a program. However, in order to acquire the design ability, learners need to find programming knowledge necessary for the assignment and organize it into a program for themselves. Current programming exercise does not distinguish learners who have no ability to organize programming knowledge into a code from others. It prevents the programming exercise from supporting learners to develop the design ability.

B. Difficulty to Train Design Ability

This paper refer to weak learners to identify the necessary programming knowledge and to organize it into a code as learners with poor design ability. In order to train design ability, it is necessary for learners to solve assignments which does not specify programming knowledge necessary to solve them, which is referred to as design dominant assignments in the paper. To solve design dominant assignments, learners have to identify programming knowledge to be used. Furthermore, they need to have a programming prospect, a plan to organize the programming knowledge into a code. . However, it involves a serious risk to make learners with poor design ability solve design dominant assignments. They might run into despair, because they are at a loss what to do for the assignment. On the other hand, learners who fail to acquire some kinds of programming knowledge cannot figure out a program based on the programming knowledge. Such learners do not know what they begin programming with at all, when they try assignments to train the design ability without proper supervision. It leads to the demotivation of learners.

From these reasons, supervisors must keep watching on their learners to see who have the programming prospect, when they provide learners with design dominant assignments. They should pay special attention to learners with poor design ability, to prevent the learners from losing motivation.

C. Relates Research

There are several works to determine individual supervisions for learners. Works in [1]-[3] record coding histories of learners using a Web site or plugins of IDE. The method proposed in these works figures out necessary guidance from the order for learners to write source codes or to commit errors. However, the method cannot grasp prospects in programming until learners write some amount of source codes.

A study in [4] gets reference histories to a library, using a console in a Web site. It has a problem beginner learners make no access to the library, because they cannot use the console.

A method proposed in [5][6] uses eye tracking to judge the level of learners. Since there are many learners in an actual class, the method is not feasible.

The methods in [7]-[9] grasp what program structures learners build, making them to draw flowcharts. In this method, learners must study new notations. It enforces efforts for learners unfamiliar with programming.

III. Nursing Education Using Reference History of Learners

A. Estimation of Programming Prospect from Reference History of Learners

To train the design ability, learners should try assignments without any guideline to solve them. Learners should make a plan to solve them for themselves. Supervisors should pay attention to learners engaging in the assignments to train the design ability, to prevent them from losing motivation. Supervisors should follow the learning behavior of learners, but should not give them guidelines to solve assignments. The paper refers to this kind of education as nursing education.

To facilitate nursing education, this paper proposes a method that judges whether learners work on design dominant assignment, monitoring how learners engage in them. Suppose a specific exercise site presenting all teaching contents, using the e-Learning technologies. It consists of Web pages explaining programming knowledge learners studied in the lecture classes, those illustrating sample codes to show how to use a specific piece of programming knowledge, and those presenting assignments for learners to solve. The exercise site also utilizes Web technologies such as Web 2.0. When learners visit any page of the exercise site, it records the user ID, the visiting time, the leaving time of the visitor, along with the ID of the Web page. Learners with programming prospects would refer to programming knowledge they have studied in assignments on the exercise site, when they solve the design dominant assignment. The method records their reference history. On the contrary, learners without any prospect would take a different reference behavior. The method provides supervisors with tailored guidelines to be presented to each learner, figuring out whether he or she has programming prospects from the reference history.

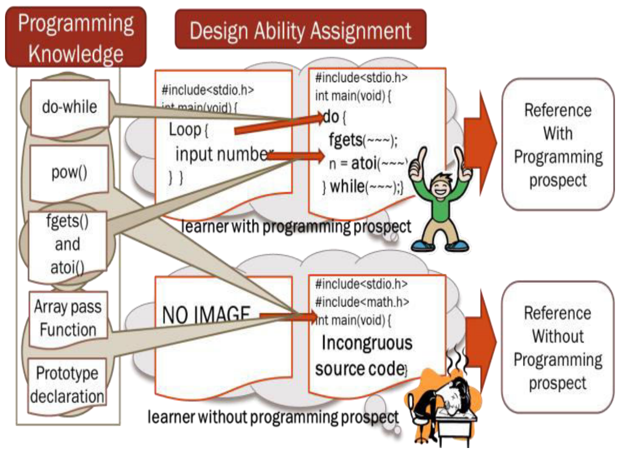


Fig1. Difference of reference action by programming prospect of learners

Fig.1 shows the reference behavior the proposed method focus on. After learners study a specific piece of programming knowledge in lecture classes, the method provides them with basic assignments in which they are expected to use the piece of programming knowledge. The method refers these assignments as base knowledge assignments. After they finish base knowledge assignments, the method make learners engage in design dominant assignments. During the engagement, learners need to recall the basic programming knowledge and its usage. They would see the base knowledge assignment they solved to understand the basic programming knowledge, as well as Web pages explaining the basic programming knowledge. Learners with programming prospects consider what kind of programming knowledge is necessary to solve the assignment, referring to the relevant pages on the exercise cite. On the other hand, learner without any programming prospect cannot grasp necessary programming knowledge. They would consult Web pages and review past assignments haphazardly, because they do not know which page they should refer to. Examining the reference history, the method determines whether learners have a programming prospect.

B. Difference in Reference by Programming Prosp-ects

Learners without programming prospect would refer to Web pages that seem to them helpful haphazardly. It increases the number of referenced pages per unit time. They also have tendency to consult pages in the sequential order, when they are searching any helpful page without any cue. Indeed, in the exercise site, a series of pages explains a set of programming knowledge corresponding to a specific teaching unit. Some adjacent pages correspond to a topic in programming in their contents, but others do not. Learners without programming prospect would visit pages in sequential order, regardless of the relevance of contents in adjacent pages, because their searching way is haphazard The proposed method pays a special attention to reference to adjacent pages, because it may be appearance of a haphazard search in high possibility. The paper refers to the movement between adjacent pages as adjacent page movement. Adjacent page movement is the indicator of haphazard search of pages.

Learners with programming prospects do not search pages in a haphazard way. They know which pages are beneficial to them, because of the programming prospect. They take pinpointed visits to pages that provide useful information for them, when they refer to page during programming. The number of referenced page per unit time is small in their reference behavior. On the other hand, some of them do not refer to any page, because they can organize programming knowledge into codes inside their mind. It is consider that they refer to almost no page per unit time.

C. Discriminating Reference Behavior

Some learners hit on ideas to organize programming knowledge into a code from the reference to Web pages. At that time, they start to write their source codes to express the ideas. When they finish the expression, they usually compile and execute their source codes to see the correctness of the ideas. If we regard the sequence of the operations as one consideration task in programming, the compilation of a source code is the finishing point of the consideration task. We can expect the reference behavior to pages in the exercise cite in the duration just before the compilation provides a key feature of each learner. It is possible to derive a guidance customized to each learner from his or her reference history in the duration.

The method extracts the reference history of *t* seconds going backward from the time of the compilation, when learners compile their source codes during exercise classes. To make the difference explained in section 3.B, the method extracts the following explanatory variables from the reference history. Suppose n is the number of pages in the reference history. Let w_i be the length of the reference time of i-th page, where i is an integer from 1 to n.Note the reference time of the page is calculated going back to the starting time of its reference, even if the reference starts more than *t* seconds in advance of the compilation.

- the number of referenced pages:n
- the average of referenced pages:
$$\overline{\mathrm{w}} = \frac{w_1 + w_2 + \cdots + w_n}{n}$$
- the dispersion of referenced pages:
$$\sigma^2 = \frac{(w_1 - \overline{\mathrm{w}})^2 + (w_2 - \overline{\mathrm{w}})^2 + \cdots + (w_n - \overline{\mathrm{w}})^2}{n}$$
- the number of adjacent page movement:m

In the reference behavior with a programming prospect, both n and m are small. $\overline{w}$ is large, because learners read useful pages using a lot of time. On the contrary, in the reference behavior with no programming prospect,$n$and m are large. It has either small $\overline{w}$ or largeσ^2, because of rough browsing of various pages.

It is possible to determine whether a learner has a programming prospect with a trained classifier which takes the explanatory variables.

D. Guidance Based on Transition of Programming Prospect

The method shown in 3.C determines whether a specific learner writes the source code with a programming prospect when he or she compiles it. Let us consider the transition of programming prospects the learner has followed. Fig.2 shows an example of the difference in the transition of programming prospects. The example shows canvas states produced by learners trying to draw an Othello board using library functions of the turtle graphics. The left side in Fig.2 shows the transition of a learner who writes a source code with programming prospects. It is considered that the learner refers to pages on the exercise site to write the source code in the following way

(1) Without any programming prospect, the learner looks for helpful programming knowledge to know how to draw squares.

(2) The learner gets a piece of programming knowledge to draw a line. Here, he catches a programming prospect.

(3) The knowledge is not sufficient to draw an Othello board. Since he loses the programming prospect, he resume a haphazard search, looking for a farther programming knowledge.

(4) He catches a programming prospect, again. He understands how to draw an Othello board.

Though the learner catches his programming prospect in duration (2), he loses it in duration (3) in the left side of Fig.2 Since he has caught a programming prospect once, he seems to search new programming knowledge in duration (3), expecting to find a cue to write the target code through the examination of various pages. Although he has no programming prospect there, we can expect he writes a code, finding a good page. Since he is improving the design ability, his supervisor had better to keep watching him without providing any guidance. It is a key approach in the nursing education.

On the other hand, a learner who is working without programming prospect for a long time continues a haphazard search, as it is shown in the right side of Fig.2. He makes no apparent progress in the programming for the assignment, which might make him desperate. Tired of the depression, he might quit learning programming. It is necessary to provide appropriate supervision for a learner who has never taken any reference behavior with a programming prospect in spite that a long time has passed from the beginning of solving the assignment

Consequently, supervisors can achieve the nursing education, examining not only the current programming prospect but also the transition of programming prospects of learners. The can choose teaching ways tailored to individual learners.

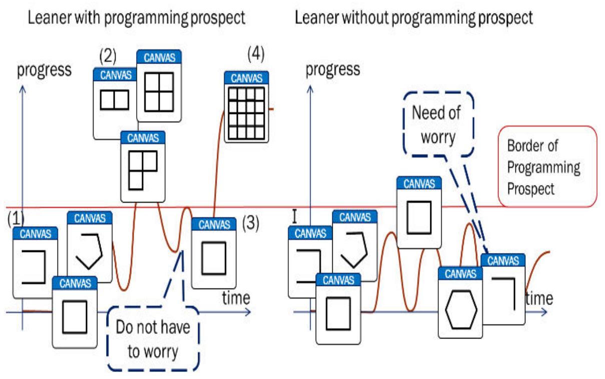

Leaner with programming prospect
Leaner without programming prospect
progress
progress
(1)
(2)
(3)
(4)
CANVAS
time
time
Do not have to worry
Need of worry
Border of Programming Prospect

Fig2. The presence of guidance by transition of programming prospect

IV. Experiment

A. System Overview

Fig.3 shows the system overview developed to confirm the effectiveness of the proposed method. Using the Web technologies, browsers of learners record reference operations to Web pages in the exercises site. The browsers send them to the system, which stores them as reference histories of individual learners. Terminal editors with which earners write source codes to solve assignments send their compilation time to the server. The system stores the reference histories in a database. Using the reference histories, the system examines programming prospects of learners whenever they compile their source codes. In addition to that, the system successively stores compiled source codes while learners engage in assignments. Supervisor can manually confirm the development of source codes of individual learners.

B. Outline of Experiment

Using the system, we have conducted an experiment for 74 learners consisting of sophomores and seniors in Danang Education University, Vietnum. They have finished learning an introductory programming course of C language. In the experiment, the learners have tried turtle graphics programming with C language. We prepare our own proprietary function library realizing the turtle graphics[10]. The exercises site has a reference manual of the library function Fig.4 shows the example of the reference manual.

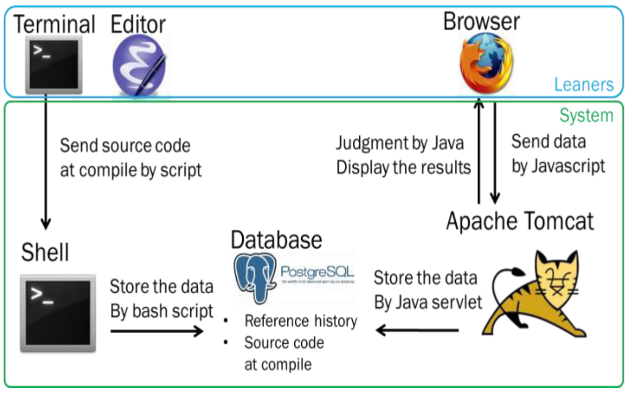
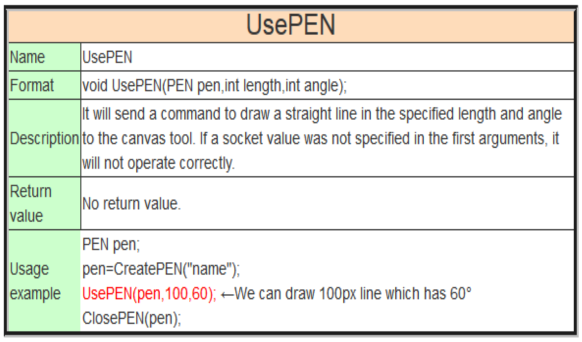

Terminal
Editor
Browser
Leaners
System
Send source code at compile by script
Judgment by Java Display the results
Send data by Javascript
Apache Tomcat
Shell
Database
PostgreSQL
Store the data By bash script
Reference history
Source code at compile
Store the data By Java servlet

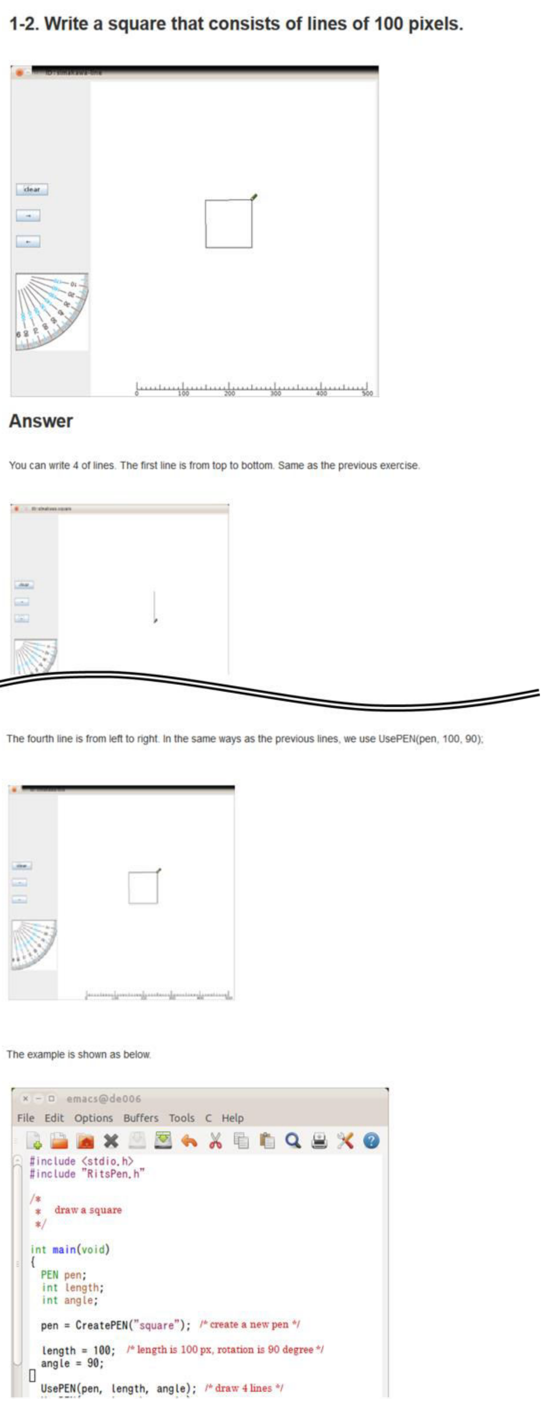
Fig3. System configuration diagram



UsePEN	
Name	UsePEN
Format	void UsePEN(PEN pen,int length,int angle);
Description	It will send a command to draw a straight line in the specified length and angle to the canvas tool. If a socket value was not specified in the first arguments, it will not operate correctly.
Return value	No return value.

Usage example	PEN pen;
	pen=CreatePEN("name");
	UsePEN(pen,100,60); ←We can draw 100px line which has 60°
	ClosePEN(pen);

Fig4. Example of a function reference



1-2. Write a square that consists of lines of 100 pixels.

Answer

You can write 4 of lines. The first line is from top to bottom. Same as the previous exercise.

The fourth line is from left to right. In the same ways as the previous lines, we use UsePEN(pen, 100, 90);

The example is shown as below.

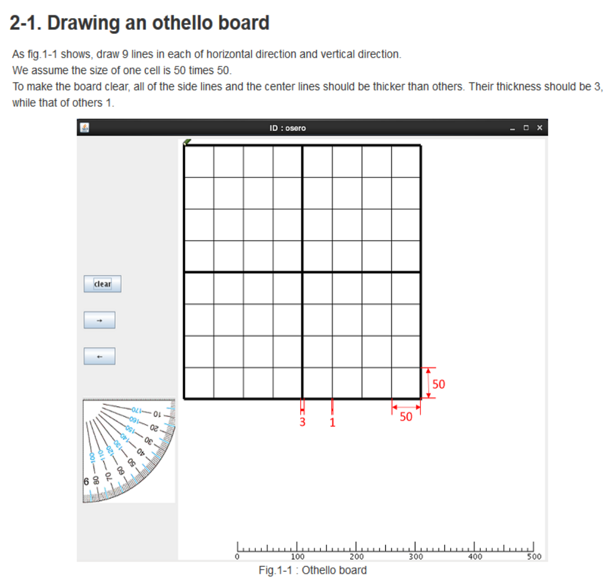
```
#include <stdio.h>
#include "RitsPen.h"

/*
 *   draw a square
 */

int main(void)
{
  PEN pen;
  int length;
  int angle;

  pen = CreatePEN("square");  /* create a new pen */

  length = 100;   /* length is 100 px, rotation is 90 degree */
  angle = 90;

  UsePEN(pen, length, angle);  /* draw 4 lines */
```

Fig5. Base knowledge assignment to make a square

The experiment aims at verifying whether the learners refer programming knowledge in the exercise site when they solve design dominant assignments. At the beginning of the experiment, the learners engage in 4 base knowledge assignments, which enforce them to make manual copies of model codes demonstrating how to use each library function. Learners debug copied source codes until they run properly, reading the reference manual. One of the model codes indicates a way to draw a square, repeating a line drawing 4 times with a for statement. Another demonstrates to draw lines with various colors. Fig.5 shows a screen shot of the base knowledge assignments to draw a square, presented on the exercise site.

The learners get programming knowledge, while they actually write source codes using each library function in the base knowledge assignments. After they finish the base knowledge assignments, we impose design dominant assignments on them. The design dominant assignments require them to write the Othello board depicted in Fig.6.

The design dominant assignment requests to make the 4 side lines thicker than the others. The learners have to write a code using the function to draw lines, specifying the length and the angle of the line. They also need the function to change the thickness of lines. However, learners need not to use the function to change the color of lines. In addition, this assignment has high relevance to the base knowledge assignment to draw a square, while no relevance to what draws lines in different colors. Sentences of the design dominant assignment do not tell the programming knowledge necessary to draw the Othello board. The learners need to consider how to organize the programming knowledge they have learned into a code.



2-1. Drawing an othello board

As fig.1-1 shows, draw 9 lines in each of horizontal direction and vertical direction.
We assume the size of one cell is 50 times 50.
To make the board clear, all of the side lines and the center lines should be thicker than others. Their thickness should be 3, while that of others 1.

Fig6. Design dominant assignment to draw the board of Othello

C. *Reference action against exercise site*

In design dominant assignments, learners without programming prospect search Web pages haphazardly, expecting to find hints in the reference manual of the function library or the base knowledge assignment they have engaged. Since they have no programming prospect, it is considered that the learners searching some hints read statements of Web pages not randomly, but sequentially. It makes them to take adjacent page movement. For example, they would read over explanations of library functions in the reference manual in the written order, or make accesses to pages of the base knowledge assignments one by one.

Meanwhile, learners with programming prospect would make pinpointed accesses to related Web pages. For example, they would inspect only the Web pages relevant to the design dominant assignment. The Web pages include what explains the library function to draw a line called along with arguments to specify the length and the angle of the line, and the library function to change the thickness of a line. Another relevant Web page shows the model code demonstrating a way to draw a square.

The proposed method can judge the presence of a programming prospect from reference history just before the learners compile their source codes.

V. Evaluation

A. *Difference in Source Codes*

As a result of the experiments, 15 learners out of 74 proceed to the design dominant assignment, after they have finished all of the base knowledge assignments. We analyze reference histories and source codes of the 15 learners, because the method targets learners who have finished acquiring the base programming knowledge.

We manually examine programming prospects for 207 source codes compiled while the learners engage in the design dominant assignment as it is shown in Fig.7. The followings are the criteria to examine a programming prospect of a specific learner.

- A learner with programming prospect draws more than one squares connecting with each other.
- A learner without programming prospect draws no square or a single square.

The criteria come from the consideration that a programming prospect would affect how they write a source code to draw the Othello board from the source code to draw a square in the base knowledge assignment. The examination based on the criteria divides source codes into 74 implying a programming prospect and 133 implying no programming prospect.

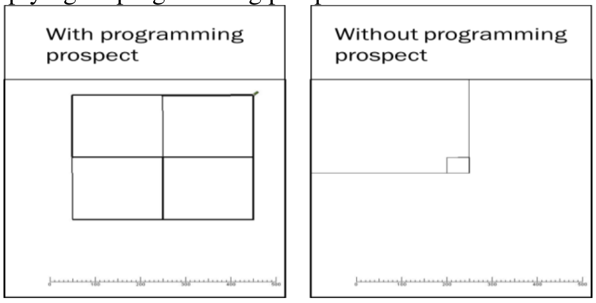


Fig7. Difference in execution of source codes with/without programming prospect

B. *Programming Prospect Derived from Reference History.*

The system extracts reference histories in a *t* second duration before the compilation. We calculate the following explanatory variables in each reference history:

- the number of referenced pages
- the average of the reference time for the pages
- the deviation of the reference time for the pages
- the number of adjacent page movement

For each reference history, a classifier based on the Support Vector Machine examines programming prospects from values of the explanatory variables. To train the classifier, we use the manual examination results explained in section 5.A.We apply the *k*-hold cross-validation, to evaluate the effectiveness of the proposed method.Table.1 shows the result of the *k*-cross-validation, where *t* and *k* are 600 second and 207, respectively.We set *k* to 207, because the number of the reference histories is 207.

The accuracy rate is calculated with the number of classifier outputs identical with the manual examination over the number of whole outputs, which is 0.78 in the experiment. Table.2 shows the recall, the precision and the F-measure.

Consequently, the method can judge programming prospect of learners from reference histories of learners.

Table1. Judgment result

		result of manual examination	
		without programming prospect	with programming prospect
output of classifier	without programming prospect	105	18
	with programming prospect	28	56

Table2. Recall, Precision and F-measure

	Recall	Precision	F-value
without programming prospect	0.79	0.85	0.82
with programming prospect	0.76	0.67	0.71

C. *Extracting the number of seconds of reference history*

We set time length of reference history *t* to 600 second in section 5.B, but the value of *t* may affect the result. We figure out the accuracy rate, increasing the value of *t* every 100 from 100 to 1800. Fig.8 shows the transition of the accuracy rate for *t*.

In this experiment, the accuracy rate gets higher as t increases, when $t \leq 700$.When $t \geq 700$, the accuracy rate decreases, as $t$increases.It tells the accuracy rate takes a peak value at a certain value of t.When t is far smaller than 700, the reference history contains too small number of references to represent difference a programming prospect brings. Because of it, the accuracy rate is small. As ***t*** gets larger, the difference by a programming prospect become vivid, which increases the accuracy rate.However, once ***t*** goes over 700, the reference history contains past references irrelevant to the current programming prospect. It confuses features brought by a programming prospect. Irrelevant noise decreases the accuracy rate.

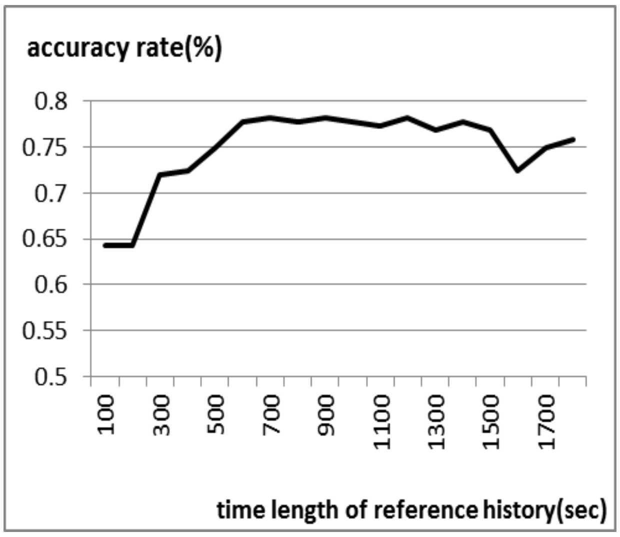


Fig8. Accuracy rate according to the length of reference histories

D. Estimation of the learners state

Some leaner catches a programming prospect, while others miss their programming prospect. It is worthy to examine programming prospects for each learner in the experiment. From the viewpoint of nursing education, we should not give immediate guidance to learners who have found any programming prospect. Meanwhile, we should give a high priority to learners who have never found any programming prospect in the supervision. A method discriminating learners brings significant benefits to the nursing education.

We regard learners who have taken any programming behavior as with programming prospects as improving learners. In the experiment, we identify 8 improving learners through manual examination. On the other hand, we regard learners failing to have programming prospect for a long time as hovering learners. There are 6 hovering learners in the experiment. Table.3 shows the discrimination result.

Table.4 shows the recall, the precision, and the F-measure in the discrimination.

From this result, the method can detect hovering learners perfectly. Improving learners may take reference behavior with programming prospect in some cases, but that without programming prospect in others. However, hovering learners take only behavior without programming prospect through the experiment. Different features in reference behavior do not affect the discrimination as noise. It implies that supervisors should determine what guidance they provide for learners, not from programming prospects in a short time, but from its transition for a long time.

Table3. Judgment result

		result of manual examination	
		without programming prospect	with programming prospect
output of classifier	hovering learners	7	1
	improving learners	0	6

Table4. Recall, Precision and F-measure

	Recall	Precision	F-value
hovering learners	1.00	0.86	0.92
improving learners	0.88	1.00	0.93

VI. Conclusion

This paper has proposed a method to evaluate programming prospects learners have when they organize programming knowledge into programs from reference histories. This method saves reference history in exercise site when learners solve the design dominant assignments. A classifier discriminates reference histories with programming prospects from others. The discrimination enables supervisors to determine what guidance they provide for individual learners at a loss to work on the design dominant assignments.

We verify the method can find programming prospects of learners in an experiment. The recall and the precision indicates the usefulness of this method. Using the method, supervisors can grasp programming prospects, which allows them to achieve the nursing education.

In the future, we apply the method to learners engaging in other design dominant assignments. In addition, we also examine educational effects of guidance based on programming prospects

References

[1] Vihavainen, Arto and Helminen, Juha and Ihantola, Petri, “How Novices Tackle Their First Lines of Code in an IDE: Analysis of Programming Session Traces”, Proceedings of the 14th Koli Calling International Conference on Computing Education Research, Koli Calling '14, Koli, Finland, 2014, pp109--116

[2] Tabanao, Emily S. and Rodrigo, Ma. Mercedes T. and Jadud, Matthew C, “Predicting At-risk Novice Java Programmers Through the Analysis of Online Protocols”, Proceedings of the Seventh International Workshop on Computing Education Research, ICER '11, Providence, Rhode Island, USA, 2011, pp85—92

[3] Piech, Chris and Sahami, Mehran and Koller, Daphne and Cooper, Steve and Blikstein, Paulo, “Modeling How Students Learn to Program”, Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, SIGCSE '12, Raleigh, North Carolina, USA, 2012, pp153—160

[4] Helminen, Juha and Ihantola, Petri and Karavirta, Ville, “Recording and Analyzing In-browser Programming Sessions” Proceedings of the 13th Koli Calling International Conference on Computing Education Research, Koli Calling '13, Koli, Finland, 2013, pp13-22

[5] Busjahn, Teresa and Schulte, Carsten and Sharif, Bonita and Simon and Begel, Andrew and Hansen, Michael and Bednarik, Roman and Orlov, Paul and Ihantola, Petri and Shchekotova, Galina and Antropova, Maria, “Eye Tracking in Computing Education” Proceedings of the Tenth Annual Conference on International Computing Education Research, ICER '14, Glasgow, Scotland, United Kingdom, 2014, pp3—10

[6] Bednarik, Roman and Tukiainen, Markku, “An Eye-tracking Methodology for Characterizing Program Comprehension Processes”, Proceedings of the 2006 Symposium on Eye Tracking Research & Applications, ETRA '06, 8, San Diego, California, 2006, pp.125—132

[7] S. Chen and S. Morris, “Iconic programming for flowcharts, java, turing, etc”, SIGCSE Bull., 37, 3, New York, NY, USA, 2005, pp.104--107

[8] C. Cilliers, A. Calitz, and J. Greyling, “The Effect of Integrating an Iconic Programming Notation into CS1”,ITiCSE ’05,New York, NY, USA, 2005 pp.108--112

[9] T. Watts, “The SFC Editor a Graphical Tool for Algorithm Development”, J. Comput. Small Coll., 20, 2, USA, 2004, pp.73—85

[10] Kohei. Tanigawa, Fumiko Harada, and Hiromitsu Shimakawa, “Detecting Learning Patterns during Exercise from Function Call Logs”, International Journal of Advanced Computer Science, 1, 1, Japan, 2011, pp.30--35

AUTHOR'S PROFILE

Yoko Itado
I Received B.E from Ritsumeikan University in 2013. She advanced Graduate School of Ritsumeikan University. She engages in the research on data engineering. She is member of IPSJ.

PHUONG Dinh Dong
Dinh Dong Phuong received Bachelor of Mathematics and Information science in 2001 in Danang Education University. Received Master of Information Engineering in Ritsumeikan University in 2008, Japan. From 2009, a Ph.D candidate at Ritsumeikan, interested in Education Engineering

Yusuke Kajiwara
He received M.E degrees from Kanazawa Univ in 2011, and his Ph.D degree from Graduate School of Natural Science and Technology, Kanazawa Univ, Japan, in 2013.Currently, he is an assistant professor of Information Science and Technology, Ritsumeikan Univ., Japan.

Hiromitsu Shimakawa
Prof. Hiromitsu Shimakawa received Ph.D degree from Kyoto Univ. in 1999. He joined Ritsumeikan Univ. in 2002. Currently, he is a professor in Ritsumeikan Univ. His research interests include data engineering, usability, and integration of psychology with IT. He is a member of IEEE and ACM.