Promoting Acquisition of Programming Skills by Reuse of Active Diagrams

Hiroshi Taguchi †  Fumiko Harada ‡  Hiromitsu Shimakawa ‡
† Graduate School of Science and Engineering, Ritsumeikan University
‡ Department of Information Science and Engineering, Ritsumeikan University
1-1-1 Noji-Higashi, Kusatsu, Shiga, Japan
hiro@de.is.ritsumei.ac.jp, harada@cs.ritsumei.ac.jp, simakawa@is.ritsumei.ac.jp

Abstract

This paper proposes a method to find clues to an understanding of C-language programming techniques from active diagrams which a teacher drew extemporaneously. An active diagram is a diagram which moves as time advances. A teacher often explains a programming technique with the movement of an active diagram. The method records active diagrams in a form that they can be reproduced afterward. The method collects also several kinds of data to grasp contents and effects of the active diagrams. The method picks out clues from the active diagrams from the data. Finally, the method provides the clues for all students eager to acquire programming skills. If they reproduce the active diagrams, they can learn effectively. We have developed data collection systems to realize the method. We plan to verify effectiveness of the method by application of the systems to an actual university course.

1. Introduction

The lack of information technology engineers who carry an advanced information society is a serious problem today. The problem demands high-quality information education from universities to breed a student an information engineer. In response, information science department of many universities provides programming courses. However, in the present state of affairs, it is often the case that a considerable number of students do not acquire programming skills when the course is over. One of the reasons is that a teacher cannot get round to provide remedial education for students who have come to a deadlock. To solve this problem, it is supposed to be effective to give know-how of students who have already acquired programming skills to other students. The know-how given to students will promote acquisition of programming skills effectively.

A teacher often draws diagrams while he explains to teach programming techniques lucidly. We think that some of these diagrams are factors in acquisition. This paper proposes a method to reuse effective diagrams which a teacher drew extemporaneously to assist students in acquisition of C-language programming skills. The method records diagrams in classes first. The method collects also several kinds of data to grasp contents and effects of each diagram. Next the method picks out diagrams which promote student’s understanding among collected diagrams based on the data. The method provides the diagrams to other students. A student who aims to acquire programming skills can learn them effectively by referring the diagrams.

Section 2 describes factors in acquisition of programming skills. Section 3 explains the method to reuse effective diagrams, and Section 4 describes implementation of data collection systems to realize the method. Section 5 compares the method with related studies. Finally, the conclusions are given in the final section.

2. Factors in acquisition of programming skills

2.1. How to learn C-language programming

The C language is a programming language that most of prospective information technology engineers learn first. According to [4], fundamental contents of the C-language programming are roughly divided into eight units, as shown in Figure 1. A student should learn through each unit first. A student who learned all units can write simple programs, but cannot write complex ones that solve general problems, e.g. data structures and algorithms. The reason is that he does not acquire a skill to combine multiple units yet.

The combining skill is a must to write programs that solve general problems. Mastering the combining skill is infinitely harder for a student than mastering one fundamental unit. Therefore, there are many students who meet with a setback in this learning process. To solve the issue, effective methods which support learning the skill are required.
2.2. Clues to an understanding of programming techniques

Programming courses in a university are made up of two kinds of class. One is a lecture class to get knowledge of programming techniques. The other is an exercise class to receive practical training of programming techniques which a student learned in the lecture class. The time until mastery of a new technique which a student learns in a lecture class differs in individuals. The time varies considerably from person to person in a case of techniques which need combining multiple units. Once a student can use a new technique, he is frequently able to make good use of it afterward.

A student who aims to acquire a new programming technique receives explanations from a teacher and others. There are a wide variety of ways to explain some technique. Ways of explanation are roughly divided into an explanation about concepts, an explanation with concrete examples, an explanation with source codes, and so on. Since an instructor can make a choice from among diverse expressions, examples, source codes, there are innumerable kinds of explanation. The more a technique is important, the more repeatedly an instructor explains it. An instructor often changes more or less a way. It is thought that some explanations which promote student’s understanding exist among innumerable kinds. Therefore, the reason why the time until mastery differs is because such explanations are different among students. In this paper, we refer to explanations that promote student’s understanding as clues to an understanding.

We have proposed a method to find slides which lead a student into an understanding among those a teacher uses in a lecture class, and applied it to actual university courses[8]. The results have proved that we can pick out slides effective to improve a student’s understanding. Accordingly, if clues to an understanding are found from teaching mediums other than slides, students will be assisted more effectively.

2.3. Active diagrams as effective teaching mediums

Two processes of understanding are needed to write a program which solves a general problem. First, a student has to understand the transaction that wants to be realized in a program. Second, he has to realize it by combining plural programming techniques. These are very hard for beginners of programming. A teacher often uses diagrams to teach the skills lucidly. A teacher represents each component of a transaction as a diagram and explains the transaction with their movement. In this paper, we refer to a diagram which moves as time advances as an active diagram. An explanation with an active diagram promotes student’s understanding than an explanation with a diagram which does not have movement, as shown in Figure 2.

In a lecture class, a teacher shows slides and examples of source codes on a display and explains them orally. In addition, he supplements the explanation with active diagrams. On the other hand, in an exercise class, a teacher provides personal coaching by showing active diagrams on paper or a display. It is highly likely that there are clues to an understanding among these explanations with active diagrams. However, a student cannot note down movement of an active diagram shown extemporaneously. If active diagrams are automatically recorded in a form that a student can reproduce afterward, and if clues are found from these
active diagrams, they will be of use to support many students.

3. Support for an understanding by reuse of active diagrams

3.1. Collection of active diagrams

The first step in finding clues to an understanding is to collect active diagrams which a teacher showed in class. It is necessary to collect several kinds of data to grasp contents and effects of each active diagram. This method has an assumption that a teacher uses presentation software, e.g. Microsoft Office PowerPoint, and draws an active diagram on a display as a supplementary explanation in a lecture class. On the other hand, it has an assumption that a teacher draws an active diagram on student’s computer when he provides personal coaching in an exercise class.

The method records movement of a diagram which a teacher draws on computer desktops in both classes as time series data. It records a screenshot of the desktop as image data at the same time. Reproduction of an active diagram is enabled by putting these together.

Information to clarify a programming technique corresponding to each active diagram is indispensable. In a lecture class, the method identifies a slide which is shown when a teacher draws an active diagram. In an exercise class, it identifies an exercise which a student is doing when a teacher provides personal coaching with an active diagram.

Furthermore, the method collects responses from students when a teacher explains with an active diagram. Whenever a student has some feelings toward teacher’s explanation in a lecture, he chooses an option which is nearest to his feelings from the following four options with the button terminal[7] as shown in Figure 3:

- **Got it!** Since I could understand the present explanation, please go ahead with a lecture.
- **Ah-ha.** Since I was interested in the present explanation, please explain it in detail.
- **Don’t understand!** Since I could not understand the present explanation, please give another explanation.
- **Wait a minute!** Since I could not understand the present explanation, please give the same explanation again.

The method records which option a student chooses and when he responds. The method identifies the same explanation again.

The method assesses usefulness of an active diagram on the basis of programming exercise results. In the method, about 10 to 20 evaluation criteria, e.g. “Is processing to open a file for reading realized?”, are set for each exercise in advance to evaluate a source code. A teacher evaluates a source code by judging whether it satisfies each criterion.

The method collects above-mentioned data about all active diagrams which a teacher drew in class extemporaneously. And the method can grasp contents and effects of each active diagram by unifying the management of these data.

3.2. Identification of programming techniques corresponding to each diagram

Since there are different clues to an understanding for each programming technique, it is necessary to identify a technique corresponding to each active diagram. The method sets corresponding techniques for each slide and each exercise previously.

Contents of a class which provides data structures and algorithms are divided into various technical items. When a single-linked list is taken for an example, technical items are “making up a single-linked list”, “inserting a node in a single-linked list”, and “removing a node in a single-linked list”. There is a one-many relation between slides/exercises and technical items. Each slide and each exercise is also corresponded to several fundamental components of a programming language which are necessary to realize the contents, as shown in Figure 4. Examples in the C language are “if-else statement”, “declaration of a pointer”, and “referring to a member of a structure”.

In the method, a teacher sets technical items and fundamental components corresponding to each slide and each exercise previously. In a lecture class, an active diagram inherits technical items and fundamental components corresponding to a slide which is shown when a teacher draws it. In an exercise class, an active diagram inherits ones corresponding to an exercise which a student is addressing.
Therefore, programming techniques corresponding to each active diagram are identified with accuracy.

3.3. Picking out diagrams which are likely to be clues

If a student can write a program with a technical item corresponding to an active diagram after he was given the diagram, it is highly likely that the diagram is a clue to an understanding. The method picks out diagrams which are likely to be clues based on student’s responses and exercise results, as shown in Figure 5. Since there are exercises that are connected with some technical items, it is necessary to judge understanding level of each technical item on the basis of exercise results. For this reason, a teacher sets technical items and fundamental components corresponding for each evaluation criteria in advance.

About an active diagram which was drawn in a lecture class, the method analyzes results of an exercise which the student was doing when a teacher provides personal coaching with the active diagram. It focuses on results of criteria related to technical items corresponding to the active diagram. If his results are good and he replied that the coaching is useful, it considers that the active diagram is likely to be clues.

3.4. Sifting out true clues

It is thought that the picked out active diagrams are capable of promoting understanding for other students who aim to acquire the same programming technique. Therefore, the method reuses them to support other students. However, it is not clear whether they will be useful for other students at this point in time. The method sifts out true clues by opening them to other students.

The method sorts these active diagrams by corresponding technical item, and opens them to other students temporarily in a state of reproducible. A student who aims to acquire a certain programming technique can reproduce any of them optionally. Then the method collects student’s response to an active diagram that he reproduced. A student replies whether it was effective for him. The method collects also results of an exercise that a student solved after he reproduces it. The method determines usefulness of an active diagram when a certain students reproduce it. In concrete terms, the method calculates the percentage of students who replied that it is effective and has good exercise results in them. The method identifies the percentage with its usefulness. If the usefulness is more than a certain threshold, the method considers the active diagram is likely to be clues.
a true clue to an understanding. In this instance, the active diagram is opened to other students continuously with its usefulness. If the usefulness is not over the threshold, it is not opened after that.

In addition, the method shows more useful information with an active diagram in order to support student’s understanding. When a student meets with a setback in learning a data structure or algorithm, it may be because he does not understand some fundamental components of a programming language. In this instance, he will not understand the learning contents even if he reviews it with a clue to an understanding. The method shows also fundamental components corresponding to each clue together as a further lead to an understanding.

However, every fundamental component corresponding to a clue is not important to student’s understanding of a technical item corresponding to it. Therefore, the method identifies really important fundamental components based on results of opening. First, the method divides students who reproduce a specific active diagram into two groups. One group is composed of students who could realize a technical item corresponding to it after he reproduces it. Another group is composed of students who could not realize. Next, the method compares both groups about results of evaluation criteria related to fundamental components corresponding to it. The method finds evaluation criteria which satisfy the former but does not the latter. The method considers fundamental components corresponding to them are really important components among every component related to the active diagram. Those active diagram is opened to other students with these fundamental components, as shown in Figure 5. The method can reuse active diagrams which a teacher drew in class extemporaneously to support other students effectively.

4. Implementation of data collection systems

We have developed three data collection systems to grasp contents and effects of active diagrams. These systems are developed on the assumption that they are used in C programming courses of a university. By combining data which these systems collect, the proposed method is realized.

4.1. Guidance sheet

The guidance sheet is software to record an active diagram which a teacher draws on computer desktop. Figure 6 shows the user interface of the guidance sheet. A teacher can draw a diagram freely with a pointing device on the guidance sheet. The guidance sheet records a process of drawing and a screenshot of desktop. The guidance sheet considers a diagram is a whole series of points. So, the guidance sheet records the following three kinds of data of each point in chronological order:

- The coordinate on computer desktop of this point.
- The relative time when this point was recorded after the guidance sheet was started.
- The flag which indicates whether a button of a pointing device was pressed when this point was recorded.

When a button is first pressed after the guidance sheet was started, the guidance sheet records the first point. While the button is pressed, the guidance sheet records a point at regular time intervals. The flag of these points is 1. When the button is released, the guidance sheet records the point and breaks recording. The flag of the point is 0. If the button is pressed again, the guidance sheet resumes recording. When the guidance sheet is exited, recording is finished. A whole series of points that were recorded until the guidance sheet is finished are one active diagram.

An active diagram which a teacher draws on the guidance sheet is reproduced by displaying lines connecting consecutive two points in order of time it is recorded. However, if the flag of the former point is 0, the line is not displayed.

4.2. Lecture responses collection system

The lecture responses collection system is composed of the button terminal and the lecture contents recording tool.

The button terminal enables a student to register a response to teacher’s explanation in a lecture. The button terminal is a web application with Java Servlet, so a student can use it by a web browser on a personal digital assistant (PDA). Therefore, the button terminal does not disturb student’s activity. Figure 3 shows the interface. When a stu-
dent presses either button, the button terminal records the kind of button and when the button was pressed.

The lecture contents recording tool records the time when a teacher changes slides on presentation software in a lecture. This tool is able to record not only the time a teacher changes a slide into the following slide but also the time he changes a slide into the previous slide.

This system identifies a slide corresponding to each student’s response by combining data which both tools collect.

4.3. Programming exercise class management system

The programming exercise class management system is a web application with Java Servlet. This system has two different interfaces for students and for teachers.

The interface for students enables a student to register attendance, view exercises and submit source codes electronically. Another interface enables a teacher to check the state of attendance, check the state of source code submission, and evaluate source codes. A teacher can evaluate each source code based on the above-mentioned evaluation method while he refers to the source code.

Students and teachers can use this system with a web browser on any computer. All information about attendance and submission, and all evaluation results are recorded on a database.

5. Related work

[2] proposed a method to support programming training by analyzing compilation results and execution results of source codes. [9] proposed a method to find students who have come to a deadlock in an exercise class. Methods to evaluate source codes automatically were also proposed[1, 5]. These methods collect data only in an exercise class. They do not reuse the collected data to assist other students in learning. To collect data in both a lecture class and an exercise class and to combine these data are essential in order to arrive at solutions for assistance to students who have come to a deadlock.

Learning environments which provide several kinds of educational materials were proposed[3, 6]. There is no environment which reuses an active diagram which a teacher drew extemporaneously in class as an educational material. A student must choose a suitable material for him among all materials in these environments. However, it is difficult for a student who has come to a deadlock to choose a suitable material for him. The proposed method provides only active diagrams which are clues to an understanding. Therefore, a student can learn effectively.

6. Conclusions

In this paper, we proposed a method to support C-language programming training. The method first collects active diagrams which a teacher drew extemporaneously in class. Next, the method finds clues to an understanding from the active diagrams. Finally, the method opens the clues with useful information to other students. Through the method, a student who aims to acquire programming skills can learn them effectively by reproducing the clues.

We have implemented data collection systems to realize the method. Our future work is to apply these systems to an actual university course and to verify effectiveness of the proposed method.

References


