A Proposal for Teaching Programming Through the Four-Step Method

1Makoto Tanabe, 2Yasu Uchida, 3Makoto Sakamoto

1Department of Intelligent System Engineering, National Institute of Technology, Ube College, 2-14-1, Tokiwadai, Ube, Japan
2Department of Business Administration, National Institute of Technology, Ube College, 2-14-1, Tokiwadai, Ube, Japan
3Department of Computer Science and Systems Engineering, University of Miyazaki, 1-1 Gakuen Kibanadai-nishi, Miyazaki, 889-2192, Japan

A R T I C L E  I N F O
Article history:
Received 22 February 2015
Accepted 20 March 2015
Available online 23 April 2015

Keywords:
CS Unplugged CS Plugged Programming Four-Step Method

A B S T R A C T
We teach computer programming to students aged 17 through 18 years. In the course, a few students consider themselves to have insufficient understanding of programming or think that they are not good at programming. In response, we adopted and implemented a part of the Computer Science Unplugged (CS Unplugged) method, which is considered an effective way of teaching information science. However, although CS Unplugged has generated considerable results in motivating students to learn and in initial learning, we feel that it is not sufficiently connected to full-fledged programming languages such as C and Java. Accordingly, we propose advancing from CS Unplugged to full-fledged programming through a new Four-Step Method. In this paper, we describe the thinking and concepts behind this proposed method.

© 2015 AENSI Publisher All rights reserved.

INTRODUCTION

The course we teach has about 40 students per class, aged 17 through 18 years. Since the students’ major field of study is Management Information, they need to learn programming (Everitt, 2014) techniques. However, the results of the survey described below show that not a few students consider themselves to have insufficient understanding of programming or think that they are not good at programming. We are examining ways to improve this situation. CS Unplugged (The Unplugged Community, 2014) is a method of teaching information science without using computers, first proposed by Tim Bell of the University of Canterbury in New Zealand. While CS Unplugged is said to be effective in teaching information science (Idosaka et al., 2011), there have been concerns that its success or failure may be an effect of the skill and experience of instructors. To address this topic, we implemented CS Unplugged for a group of students of a different age group, from fifth through ninth grades, and looked at their responses. The results showed that after first making sufficient preparations the method could generate results without necessarily depending on the skill or experience of instructors. Other research underway in Japan includes a study on use of teaching aids in learning about algorithms through CS Unplugged (Manabe et al., 2013) and a study on learning the fundamentals of computer programming through a programming learning environment for beginners (Nishida et al., 2007). A study by Feaster et al (2011) concerns practice in teaching high-school students, and it has been reported to have had some, albeit limited, success.

However, at present almost no research has been conducted on advancement from CS Unplugged to full-fledged programming languages. Accordingly, we propose a new method of advancing from CS Unplugged to full-fledged programming. The proposed method begins with conducting a CS Unplugged activity, and then continues on to writing a program on the same theme and further to its abstraction in Java. Accordingly, we propose advancing from CS Unplugged to full-fledged programming through a new Four-Step Method. The proposed method consists of the following steps: Step 1, A CS Unplugged activity; Step 2, A CS Plugged activity; Step 3, Preparing pseudocode; and Step 4, Writing Java source code.

1. Background of this study:

Every year we conduct a survey following our first-term course of two 90-minute sessions per week held over 30 weeks. Here we will look at numbers of students who answered “yes” or “no” to this survey’s...
question, “Do you think you largely understand Java?”

As seen in Fig. 1, the survey’s results show that not a few students consider themselves to have insufficient understanding of programming or think that they are not good at programming. As such, there is a need to improve this situation.

![Fig. 1: Survey results on understanding of Java.](image1)

2. **CS Unplugged in practice:**

   For about 50 minutes on August 2, 2014 we used the CS Unplugged method for approximately 30 students from fifth through ninth grades as part of a summer-vacation junior science course. The activity we implemented was CS Unplugged’s Image Representation activity. Fig. 2 shows a scene from this activity, while Fig. 3 shows examples of students’ work.

3. **The Four-Step Method:**

   In this paper, we use the name CS Plugged to refer to implementing a CS Unplugged activity through a computer program. The goal is to advance to computer programming through using a computer program to conduct the work done by human beings in a CS Unplugged activity.

   Accordingly, we propose advancing from CS Unplugged to full-fledged programming through a Four-Step Method. The proposed method consists of the following steps: Step 1, A CS Unplugged activity; Step 2, A CS Plugged activity; Step 3, Preparing pseudocode; and Step 4, Writing Java source code.

   **Step 1: CS Unplugged.** First, students conduct the CS Unplugged activity as described in Previous above.

   **Step 2: CS Plugged.** Here we will look at the example of using a computer program to implement the CS Unplugged activity Image Representation. This is conducted through two activities. The first, converting the image to code, is represented in Fig. 4. When students click on squares in the grid at left to draw a picture, the image is converted instantly to code as displayed at right.

![Fig. 2: Scene from course practice.](image2)
Step 3: Preparing pseudocode. In this step, the three elements of sequence, decision, and repetition through preparation of the trace table in the preceding step are extracted and represented in pseudo-language. In Japan, the national Information Technology Engineer Examinations employ pseudo-language (Information-technology Promotion Agency Japan, 2011). A pseudo-language simulator is a type of software that makes such pseudo-language executable. In this paper, we used the
freeware pseudo-language simulator SARA (Mimura, 2014). Fig. 6 shows the code written in this step and its execution.

- Program name: Image2Code
- Function: Display function (A)

- Subscript 1 ← 0
- Subscript 1 ≤ 1
  - Previous character ← "□"
  - Count ← 0

- Subscript 2 ← 0
- Subscript 2 ≤ 4
  - Display processing (Array [Subscript 1, Subscript 2])
  - Subscript 2 ← Subscript 2 + 1

  ▲ Array [Subscript 1, Subscript 2] = Previous character
  - Count ← Count + 1
  +-----
  - Display processing (Count)
  - Display processing (",")
  - Previous character ← Array [Subscript 1, Subscript 2]
  - Count ← 1
  ▼
  - Subscript 2 ← Subscript 2 + 1

  □
  - Display processing (Count)
  - Display processing ("Line feed")
  - Subscript 1 ← Subscript 1 + 1

Tell students that the subsequent debugging process is handled by going back and examining each previous step depending on the content of the error messages.

**Step 5: Writing Java source code.** In this step students create a program by converting the pseudo-language from the previous step to Java source code.

- Java source code

```java
public class Image2Code
{
    public static void main(String args[])
    {
        char[][] image = {{'□', '■', '■', '■', '□'},
                          {'□', '□', '□', '□', '■'},
                          {'□', '■', '■', '■', '■'},
                          {'■', '□', '□', '□', '■'},
                          {'■', '□', '□', '□', '■'},};
```

Fig. 6: Sample use of pseudo-language simulator.
```java
// Iterate only number of lines in an array
for(int i = 0; i < image.length; i++) {
    char previous = '□';
    int count = 0;

    // Display the image in pixels
    for(int j = 0; j < image[i].length; j++) {
        System.out.print(image[i][j]);
    }
    System.out.print(" ");

    // Count number of adjoining pixels of same color and display in numerical form (code)
    for(int j = 0; j < image[i].length; j++) {
        if (image[i][j] == previous) {
            count++;
        } else {
            previous = image[i][j];
            System.out.print(count + ", ");
            count = 1;
        }
    }
    System.out.println(count);
}
```

---

4. Conclusion:

We have proposed a new method of advancing from CS Unplugged through the new process of CS Plugged to full-fledged computer programming languages, as a means of deepening understanding in computer programming education. We also have proposed a new Four-Step Method consisting of the following steps: Step 1, A CS Unplugged activity; Step 2, A CS Plugged activity; Step 3, Preparing pseudocode; and Step 4, Writing Java source code. Topics for the future are those of putting together detailed procedures for abstraction of pseudocode from trace table as well as continually implementing the proposed method and measuring its results. If you follow the “checklist” your paper will conform to the requirements of the publisher and facilitate a problem-free publication process.

REFERENCES


Mimura, S., 2014. Gijigengo shimyureta SARA (“SARA pseudo-language simulator”), Information
