Understanding comprehension of iterative and recursive programs with eye tracking

Arooba Aqeel, Norman Peitek, Janet Siegmund
arooba.aqeel@informatik.tu-chemnitz.de
University of Technology Chemnitz

Abstract

To increase the understanding of program comprehesion, we will conduct an eye-tracking experiment to evaluate whether iterative and recursive source code elicit different eye movement patterns. This would indicate that they have different underlying cognitive processes.

1 Introduction

For decades, researchers have been working on understanding how programmers understand and comprehend source code. However, understanding source code involves internal cognitive processes that cannot be measured directly, making it difficult for researchers to fully understand it. Fortunately, more direct measures have found their way into software-engineering research methods and eye tracking. This allows researchers to take a closer look at the cognitive process of program comprehension. Especially the discussion of how object-oriented and functional programming drive the comprehension process have long been under debate.

The imperative programming paradigm focuses on how programs work, where the code explicitly describes the control flow and the instructions that modify variable values (i.e., the program state). Iterative code follows this paradigm.

In contrast, recursive code relies on the declarative programming paradigm, which focuses on what programs should perform without describing the control flow explicitly, and can be considered as the opposite of imperative programming. Functional programming follows this paradigm.

The computational power of computers is mainly due to their ability to carry out tasks repeatedly, either through iteration or recursion. The former uses constructs such as while or for loops to implement repetitions, while recursive functions invoke themselves or other functions successively, carrying out tasks repeatedly in each call, until reaching a base case [1]

Basic Recursion like tail Recursion and Iteration are equivalent in the sense that they can solve the same kinds of problems. As we move to advanced recursive functions like multiple or mutual recursive functions we observe more differences in both types of code.

In this paper, we propose a study on the cognitive abilities of a programmer to comprehend source code that uses algorithms with either recursion or iteration.

Sulov conducted a study to identify developers' initial preference, success rate, comprehension and subsequent preference when dealing with programming tasks which could be solved using either iteration or recursion. These studies prove that students who are a novice in programming prefer iteration over recursion in most cases and prefer recursion over iteration only when there are clear indications [2]

2 Aim

To increase our understanding of program comprehension, we will conduct an eye-tracking experiment to evaluate whether iterative and recursive source code elicit different eye movement patterns, which would indicate that they have different underlying cognitive processes. As iterative code has iteration statements like “for” loops, “while” loops, or “do-while” loops as shown in Figure 1. We aim to understand the cognitive process of how a programmer's brains understand such statements. Recursion on the other hand calls a function repetitively, the recursive code must include a select statement in the definition of the function to force the function to return without giving a recursive call to itself as shown in Figure 2.

```
// ----- Iteration -----
// Method to find the factorial of a given number
int factorialUsingIteration(int n)
{
    int res = 1, i;

    // using iteration
    for (i = 2; i <= n; i++)
        res *= i;

    return res;
}

// Driver method
int main()
{

   cout << "Factorial of " << num <<
           " using Iteration is: " <<
           factorialUsingIteration(5);

    return 0;
}
```

Figure 1: Snippet of an Iterative function for finding the factorial of a given number [3]

3 Experimental

We plan to coduct a controlled eye tracking experiment. To thisend, we design a set of iterative and recursive source code snippets that we show to a group of computer-science students. There will be six snippets with one function each. We track their eye movements with a Tobii X3-120 eye tracker [4], this will help us understand whether iterative and recursive snippets elicit different eye movements patterns (e.g., in terms of number of saccades or pupil dilation).

The participants in our study will be undergraduate students of the University of Technology Chemnitz. We will have two groups of students. Students from year one are already fimiliar with the basics of Java, data types,flow control statements,arrays,structures, user functions,etc.The second group are students from third year who are more experienced with devising algorithms using iteration and recursion.

```
// ----- Recursion -----
// method to find factorial of given number
int factorialUsingRecursion(int n)
{
    if (n == 0)
        return 1;

    // recursion call
    return n * factorialUsingRecursion(n - 1);
}

// Driver method
int main()
{
    int num = 5;
    cout << "Factorial of " << num <<
            " using Recursion is: " <<
            factorialUsingRecursion(5) << endl;
    return 0;
}
```

Figure 2: Snippet of a recursive function for finding the factorial of a given number [3]

A total of twenty students will participate, ten for each group. These program snippets range from a few lines to an entire screen full of text. Programs will be complete Java classes. All students will also look at six programs in total. Two of these programs for the second group will be the same as the programs used for first year students. The remaining four programs will be comparable in length to beginner student’s programs, but will be advanced in programming methods.

Since we are interested in both novices and experienced students, it is important to determine the expertise of the participants. In order to evaluate the experience of students we will use a programming

experience questionnaire proposed by Seigmund et al.,2012 [5] right before the first measurement. This questionnaire will enable us to assess the required level of development of the participants in terms of their programming skill.

Immediately after reading each program, the participants will be asked to determine the output by naming the variable or write its value after program execution.

4 Analysis

During an eye tracking experiment three main data sets are recorded with a number of measures.
a) Behavioral data; for analyzing how well students comprehended the snippets. Two main factors are to be noted which are accuracy rate and completion time
b) Eye-tracking data: data we need for eye tracking depends on many measures. The first is the sequence of gaze fixations, each containing a gaze location and the fixation's duration (in milliseconds). The next measures are total saccades, saccades per second and saccade length. Saccade length is the average distance between the participant's two consecutive fixations in each trial. This will calculate which snippets were difficult to comprehend by participants. How many times will the participant have to revisit a loop or recursive statement. Another measure is element coverage which measures the percentage of words (i.e. source code elements) in the text that the participant will look at. This will show what were the words that were focused more by participants and hence resulted in better comprehension.
c) Pupil size / dilation: An increase in pupil size is referred to as pupil dilation, and a decrease in size is called pupil constriction. The attribute that can be derived from changes in pupil size in this case is cognitive workload [6]. fMRI studies have shown the increases in brain activity within different areas have been directly linked to pupil dilation . This will help analyze which program parts were harder to understand

5 Expected Results

In general, we expect to see changes in participants' eye movements when reading code snippets over the different trials. Participants will elicit different eye movements while reading iterative and Recursive functions. In addition, we expect that developer's comprehension abilities correlate with their experience. We expect to see minor differences in eye movement for basic recursive functions and their corresponding iterative functions. However, with advanced recursive functions and corresponding iterative functions we expect noticeable differences in eye movement.

6 Conclusions

Our study will identify the cognitive process behind program comprehension. This will help design better teaching techniques for recursion and iteration, which in return will be useful for professional implementation in the industry.

References

[1] M. Rubio-Sánchez, Introduction to Recursive Programming, vol. 1, CRC Press,, 2017.

[2] V. Sulov, "Iteration vs Recursion in Introduction to Programming Classes: An Empirical Study," Cybernetics and Information Technologies, vol. 16, no. 4, pp. 63-72, December 2016.

[3] P. Nell Dale, D. T. Joyce and C. Weems, "Object-Oriented Data Structures Using Java, 4th Edition," Jones & Bartlett Learning, 2016, pp. 166-168.

[4] T. Group, "Tobii Pro X3-120," [Online]. Available: tobiipro.com/product-listing/tobii-pro-x3-120/. [Accessed 11 March 2020].

[5] J. Siegmund, C. Kästner, J. Liebig, S. Apel and S. Hanenberg, "Measuring and modeling programming experience," Empirical Software Engineering, vol. 19, pp. 1299-1334, 2014.

[6] G. J. Siegle, S. R. Steinhauer, V. A. Stenger and R. Konecky, "Use of concurrent pupil dilation assessment to inform interpretation and analysis of fMRI data," NeuroImage, vol. 20, pp. 114-124, 2003.