Homework 07 - Poster Text

Andrew Carter and Beryl Egerter

March 10, 2013

1 Title

A Qualitative Analysis of Code Comprehension and Debugging Strategies

2 Abstract

We performed a qualitative analysis on a study consisting of four interviews of college students with varying amounts of experience beyond basic programming. In the interviews, we asked the students to perform code comprehension or debugging on four or five code samples with varying levels of difficulty. Some code samples included programming concepts or syntax unfamiliar to the students. We are interested in the different strategies the students used for exploring unfamiliar code and have qualitatively analyzed how well the strategies worked on our code samples. Strategies we saw included students delving into the details of a function or obtaining a high level summary of a function from the name. Some students even changed strategies depending on the given task. We believe evaluating these strategies is important because the strategies can affect the success or failure of programmers trying to maintain, change, or interact with code written by others. These situations often occur in industry. Our study shows that programmers may change strategy based on problem type and complexity, of which future research may need to be careful about.

3 Introduction

While many studies have looked at code comprehension strategies and many studies have looked at debugging methodology, we were interested in how a student's strategy changes between these tasks. We performed a study on code comprehension involving evaluation or debugging on different code samples. Evaluation is figuring out what a piece of code will output, while debugging is looking through a piece of code to identify the cause of a specific problem. For this poster we have qualitatively analyzed one student's response to an evaluation question and a debugging question. We found that while the student could have followed the execution of the program in both code samples, while evaluating, he followed the execution, but while debugging, he searched top down for where the problem could occur.

4 Methods

4.1 Data Collection

We designed our data collection to capture the code comprehension of a student looking at pieces of paper containing code in the programming language Python. A camera was set up to capture the hands of the student interacting with pen and paper as well as audio of anything they said out loud. At times the student was prompted by the interviewers to provide more information.

4.2 Problem 1.1

4.2.1 Description

In this evaluation question, we asked, "For this question we would like you to familiarize yourself with some Python code. Please explain to us what you think this code does."

4.2.2 Code

```
def func2(list, num):
    return func1(list, num, func4)

def func4(a, b):
    return a * b

def func1(list, num, f):
    acc = 0
    for i in list:
        acc += f(i, num)
    return acc

def main():
  print(func3([1,2,3,4]))

def func3(list):
  return func2(list, 4)

main()
```

4.2.3 Solution

This code prints out the number 40.

4.3 Problem 2.2

4.3.1 Description

In this debugging question, we asked, ”For this question we would like to have you look at some code in Python. This is the scenario: You acquired a connect 4 program from a friend. However, the friend has warned you that you can put too many pieces in a column. Determine a possible fix for this bug so that you can enjoy your connect 4 program.”

4.3.2 Code (First 13 lines)

```
#!/bin/env python3

class Board(object):
  def __init__(self, width=7, height=6):
    self.board = [[] for i in range(width)]
    self.width = 7
    self.height= 6

  def drop(self, player, column):
    if column < len(self.board):
      self.board[column].append(player)
      return True
    return False
```

4.3.3 Solution

While there could have been multiple solutions, we were looking for students to tell us to add an extra check within the drop function checking if the length of the list representing the column was full.

5 Narration of Data

ACBE3, Q1.1
ACBE3, Q2.2

6 Discussion

When following the thread of execution, the unfamiliar concepts were easier to understand. One student started from main and evaluated it by following thread of execution. Other student went function by function from the top, but ended up having to follow thread of execution.

7 Future Work

One of the students we interviewed switched things up between evaluation questions (format of code and how it affects strategies the student decides to use?)