Homework 07 - Poster Text

Andrew Carter and Beryl Egerter

March 12, 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

5.1 Narration of Q1.1

The student begins talking as soon as he is handed Question 1.1

1:55 : Alright umm, so, its got a main, so thats gonna start ummm, its going to print whatever the result of
2:00 : function 3 on 1, 2, 3, 4, some array. So lets see,
2:05 : Function 3 takes a list and returns whatever function 2 does called with list
2:10 : and some argument 4. Function 2 returns function 1
2:15 : with the same two arguments already passed to it, and
2:20 : function 4 the result of, which doesn't have any
2:25 : implied arguments, thats interesting. Umm,
2:30 : ...

The student identifies main, and then is able to quickly trace through the program through func2. When the student arrives at func2, he identifies function 4, which is in the argument list, but is confused because no arguments are being passed to the function. He is further confused, because the function arguments do not have any default values, which would indicate that func4 could be called without arguments. Although in python syntax it still would require parenthesis, which make this odd.

2:35 : thats odd, lets see, so its calling functi-, ooh,
2:40 : its calling function 1. There we go. Uhh, yes, so its calling function 1 with a list a number, the array
2:45 : and 4, and the function 4 as sort of a multiplier. A function to apply.

He then looks at function 1, is able to identify the use of f in function 1, which is what is passed as function 4. The student then works out the semantics of passing functions from how the code functions.

2:50 : Alright, so function 1 is doing the actual work here. Umm, see, it starts with some accumulator 0,
2:55 : iterates across the uh items in the list, list,
3:00 : and plus equals that function 4 applied
3:05 : to i being the item from the list and that number 4 that was included in function 3.
3:10 : So its going to essentially sum the list multiplied by 4,
3:15 : it would appear, and print that sum out.

3:20 : Yeah, its gonna take, each element 1 2 3 4 and multiply it by 4
3:25 : add that to 0 and then return the accumulator
3:30 : back up the steps. So function 1 2 3 yeah.

Once he understands passing functions, he quickly figures out what the inner function does, then unwinds the functions and determines what the overall program does.

5.2 Narration of Q2.2

The student begins talking as soon as he is handed Question 2.2

18:54: Alright so we have a board class which is gonna presumably represent the connect 4 board... um...
19:00: the constructor automatically sets the width to seven oh in this case it forces constraints um like arguments essentially ... and width is always going to be equal to seven, the height equal to six
19:15: um ... and its going to create an array of arrays or a list of lists depending on what you call it in python ... um ...
19:28: then it appears to not force the height to be six
19:32: for i in range width ... yeah that could be cause of problems ... um
19:42: i don't know if, if lists in python are dynamically allocated or not
19:49: i don't think yeah i think they are, you can keep adding to them can't you...
19:54 (I): Yes
19:55: Yeah ... so i wonder what this is doing. I guess it would be creating ... an array with at least six - er, seven secondary arrays in it
20:09: um ... and you don't necessarily [gestures] specify the height ... i guess that doesn't matter you just need to check along the way ... um ...

The student started at the top of the file with the class and relates it too the problem statement. He looks at the constructor and identifies what the global variables could be. He also talks about arrays and lists, showing confusion about how they work in Python. He finishes this section with an understanding what each of the member variables of the class is.

20:20: alright so drop. i guess that is where you drop a piece in ... if column less than self.width um self.board.append player

20:32: oh ok so its doing [starts drawing on paper] like a ... array of arrays essentially and then it just pops on a color - i don't know are they black and red? I think they are... so sorta pops on a color as they happen and that way ... oh i guess no append is going on the end isn't it ...um
21:01: so this is where you would need to do the check [points at drop function with pen]

The student has moved from the constructor to the drop function which is located right after the constructor within the code. He uses the name of the function to relate it to an action he knows occurs in a connect four game. The code of the function, in particular, the append, helps him understand the board member variable further as an array of arrays. He realizes that this is an appropriate place to fix the given bug.

21:04: if column less than self.width then append it um
21:15: You could do - you could add to that if statement? let's see ... and [writing on paper] um
21:25: column.size, is that a thing? oh no its going to be board at column is what it's going to be board at column
21:38: some sort of size operator
21:42 (I) : It's len
21:44: Ok, len ... um ... less than I have ...[mumbles] ... and board column . len less than ... height is six so it's gotta be - oh we can just do less than self.height
22:10: ...gotta keep good encapsulation...um ... yeah so this check would go right here [points at if in drop function]
22:20: and that would keep you from ... otherwise it would return false and not allow you to drop if you exceeded the height ... um ... bounds ... that would work.

After identifying that this is a good place to fix the given bug, the student finds a way to modify the code to actually fix the bug. Further evidence of the student's unfamiliarity with Python is seen by his uncertainty about the size operator. Despite this, the student comes up with appropriate pseudocode to solve the problem.

---

6 Discussion

The student uses two different strategies to solve the problem in each section. While there are differences between the code used in each problem such as one being much longer than the other, and one including a class as opposed to a number of functions, they both have pieces of code that execute. In the first problem, when the student is told to figure out what the code does, he follows the thread of execution. By following the thread of execution, when he encounters the function being passed as an argument, he is able to figure out what is occurring by following the execution. It is possible that had the student worked from top to bottom he might not have realized the connection between passing func4 and calling f, and thus taken much longer to understand what the code is doing. In the other code sample, when the student is first told the overall function of the code and a bug to fix, the student started from the top of the file and moved down until finding a likely area that could contain the bug. It would have been possible to start where the program starts executing and trace the thread of execution to where the pieces drop. In fact, by looking through the code from the top and stopping at the drop function, the student doesn't actually know if this function is ever used. The student is assuming the author of the program wrote it well. Unlike the first code sample, if the student had followed the execution of the program, it would have likely taken him much longer to look at the code where the bug could be fixed.

7 Future Work

In future work, we hope to explore strategies used by more students and how it impacts their ability and confidence in overcoming unfamiliar constructs or semantics. For instance, one of the students we interviewed switched strategies between two evaluation questions based on the perceived complexity of the code. Another started all of the questions by looking at the code from the top down, but had varying degrees of success depending on the problem. A couple of students read through the entire code base for the debugging problems, and got bogged down in trying to understand complicated functions, to the point of forgetting the initial debugging problem. We would like to explore the advantages and disadvantages of familiarizing oneself with the code base to varying degrees such as: a detailed understanding of each function, a high level understanding of the whole program, or determining where the most likely location of the bug is without regard to anything else.