AN INTEGRATED APPROACH TO INSTRUCTION IN DEBUGGING COMPUTER PROGRAMS: PRELIMINARY REPORT

Ryan Chmiel[1] and Michael C. Loui[2]

Abstract *— The purpose of this study was to demonstrate that formal training in debugging helps students learn to diagnose and remove defects in computer programs. To accomplish this goal, students completed sets of debugging exercises before coding the programming assignments in an assembly language course. Each set of exercises focused on the major topics covered in the corresponding assignment. Students also kept debugging logs as they worked on the assignments. In these logs, students recorded the source of each defect and how the defect was corrected. Student response to these exercises, in the form of surveys, has been positive and constructive.*

INTRODUCTION

Although debugging is an integral and time-consuming aspect of software development, few computing curricula offer formal training in debugging. Consequently, students must develop debugging skills on their own. It is not clear how instruction can help students improve debugging skills. Previous researchers have proposed instruction in program comprehension [1], code review [2], and peer review [2].

Statistics about code defects support the need for formal debugging training. A few defect types occur frequently in code [3]. Through training, programmers could become aware of high frequency defects. Ideally, this awareness would result in a reduction in these defects.

Lee and Wu developed an interactive tool to help novice student programmers improve their debugging skills [4]. The tool was capable of helping students identify and correct only defects related to loops, however. Our work includes a greater variety of defect types.

METHOD

We created one set of debugging exercises for each programming assignment in a course on assembly language and real-time computing. Each set of exercises contained two types of problems. For the first type of problem, students identify defects in short subroutines (about 20 instructions long) using code review techniques. For the second type, students identify and correct the defects by modifying and testing the code with a debugging tool. In each set of exercises, there are several types of defects. Most defects in a particular set are common defects—off by one, wrong jump condition, addressing mode error, type mismatch, and so on; however, more obscure defects are also included to give students practice in locating them as well.

We developed a debugging log for students to keep as they worked on the programming assignments. The log is modeled on a log proposed by Humphrey [2]. Upon encountering a new defect, the student records the following information in the log: the subroutine containing the defect, the time taken to correct the defect, the incorrect program output and/or behavior, the faulty code, and most important, the solution to the defect. By recording all defects in a log, the student can keep a list of previous defects and their solutions for use in correcting future defects.

RESULTS

In the fall of 2002, we obtained preliminary qualitative results and observed positive outcomes. Students felt that the exercises helped improve their debugging skills. Furthermore, students suggested improvements in the exercises. For example, they suggested that the number of defects per problem be decreased.

We are expanding the study during the Spring 2003 semester. We believe that these additions to the course are beneficial in improving students' debugging skills.

REFERENCES

[1] Gugerty, L. & Olson, G.M., "Comprehension Differences in Debugging by Skilled and Novice Programmers", *Empirical Studies of Programmers*, 1986, pp. 13-27

[2] Humphrey, W.S., *Introduction to the Personal Software Process*, 1997, pp. 143-149, 159-172

[3] Spohrer, J.G. & Soloway, E., "Analyzing the High Frequency Bugs in Novice Programs", *Empirical Studies of Programmers*, 1986, pp. 230-251

[4] Lee, G.C. & Wu, J.C., "Debug It: A debugging practicing system", *Computers and Education*, Vol. 32, No #2, 1999, pp. 165-197

Supported by the National Science Foundation under Grant SES-0138309 and an Architecture for Change grant from the College of Engineering at the University of Illinois at Urbana-Champaign. The opinions, findings, and conclusions of this paper are not necessarily those of the National Science Foundation or the University of Illinois.

[1] Ryan Chmiel, Dept of Electrical and Computer Eng, University of Illinois at Urbana-Champaign, 1406 W. Green St, Urbana, IL 61801, rchmiel@uiuc.edu

[2] Michael C. Loui, Dept of Electrical and Computer Eng, University of Illinois at Urbana-Champaign, 1406 W. Green St, Urbana, IL 61801, m-loui@uiuc.edu