IMPROVING DEBUGGING EDUCATION THROUGH APPLIED LEARNING*

Xiaohong (Sophie) Wang and Joshua Souders
Department of Mathematics & Computer Science
Salisbury University
1101 Camden Avenue,
Salisbury, MD 21801 USA
Telephone: (410) 677 5380
xswang@salisbury.edu

ABSTRACT

In this undergraduate research project, an extensive study on current software debugging methodologies and pedagogical approaches to debugging education in computer science curriculum is conducted. Based on the research findings, a web-based software tool is developed to assess the debugging skills for computer science students. Based on the outcome of the assessment, the tool can provide appropriate tutoring and practice exercises to target at the skill weakness of each individual. The benefits for this first-hand research and development experience in software debugging include the introduction of the debugging methodology research and education to freshmen and sophomores in computer science and the creation of a unique debugging skill assessment and tutoring software that can be adopted by undergraduate computer science education community.

INTRODUCTION

Debugging is an important skill that is both difficult for novice programmers to learn and challenging for educators to teach. McCauley, et al. [10] conducted a thorough literature review in teaching debugging skills and pointed out that difficulties and challenges still persist despite a wealth of important research on the subject and the improvements of programming tools and languages in the past twenty years. The difficulties and challenges, as also identified by Laporte, et al. [8] are lack of time and low

perception of the importance of software quality practices as the result of the constant struggle due to the ever-increasing wealth of topics to be covered in computer science programs and limited required credit hours for graduation [12]. To address all those issues within busy curriculum and limited budget requires a lot of creativity. Some institutions use quality-centered teaching approaches in all computer science courses [6]. Some use laboratory-based non-credit seminars on debugging related topics [8]. Some adopt in-class technical review, pair programming, personal software process and refactoring to teach and improve debugging skills [1, 3, 9, 13, and 14].

To further support and facilitate those innovative approaches to debugging skill teaching and provide the much needed pedagogical materials and tools, an undergraduate research project is conducted among a group of undergraduate computer science students. The objectives for this undergraduate research project are to: 1) foster software quality awareness among freshman and sophomore computer science students; 2) enrich their knowledge and experience in debugging techniques through an in-depth study on modern debugging methodologies and debugging teaching approaches; and 3) provide the opportunity to apply their findings in the development of a web-based debugging assessment, analysis and tutoring system.

The rest of this paper is structured as follows. In the next section, the importance and challenges in teaching debugging skills are discussed. In section 3, the experience of the undergraduate research project is described in details. The last section summarizes the results of the undergraduate research project.

BACKGROUND

Novice computer science students face many challenges in learning the basic skills required to design and implement programs. Thinking with abstract concepts, multitasking with problem solving, algorithm and data structure design, programming language comprehension are all new to them. Trying to master all those skills simultaneously can be overwhelming. In the meanwhile, weakness in any one of those areas can lead to inefficient programming experiences and buggy programs.

According to McCauley, et al. [10], programming is a process of using source code to build a plan to achieve a certain goal. A bug leads to the breakdown of the plan. The causes of bugs can stem from the preconceptions, misconceptions, and the fragility of knowledge and understanding of programming and programming languages.

A survey conducted by Kazemian and Howles [6] among freshman computer science students found that only small percent of their students always developed a design and a plan, static checked their work, and performed unit test consistently. The reason for not designing, checking and testing was lack of time. The students spent too much time trying to get their programs to compile and run. Their results also indicated that lack of software quality awareness, misconceptions of programming languages, and poor debugging skills are the main culprits of the inefficiency in novice programming.

Fitzgerald, et al., [2] reported their general quantitative results of a multi-institutional study of novice debuggers' skills and behaviors. The approaches used in their study are comprehensive interviews, semi-conducted interviews, questionnaires and observation during a programming task and a debugging task among a group of their students. The

data from the interviews, questionnaires and observations were used to rank the effectiveness of debugging strategies used and the noteworthiness of observations, identify unproductive behaviors and inferior processes adopted. The percentage of the bugs found was taken into consideration as well when judging the debugging performance for each student. This study confirmed many previous findings and suggests some new findings, such as an increased use of tools, online resources, and pattern matching technique, and the importance of meta-cognitive awareness in debugging performance.

Those studies and findings give valuable insights to many general novice programmer behaviors and guidance for debugging teaching. However, according to the preliminary research done by the students working on this undergraduate research project, to adopt those findings in teaching debugging skills, practical and effective supplement tools are still needed in order to fully utilize those results. In this undergraduate research project, an effort is made by incorporating those research findings into a web-based tool that can assess, analyze, tutor and practice debugging skills. Specifically, this application provides assessment of a student's debugging skills through a series of quizzes. The questions in those quizzes are designed based on those research findings and they are used to evaluate a student's knowledge and skills in the areas of programming language constructs, compilation and logic error message comprehension, source code comprehension, debugging strategy selection, etc. Based on the result of assessments, an individual student's strength and weakness in those areas can be identified. Appropriate tutorials and practices to improve those skill weaknesses are recommended.

UNDERGRADUATE RESEARCH EXPERIENCE

Objectives

The undergraduate research project described in this paper is partially supported by NSF STEM grant. One objective of this grant is to foster a community of research and engaged learning for students early in their careers. Besides this major objective, this specific undergraduate research project also strives to infuse software quality awareness and teach debugging skills as early as possible. This goal is achieved by involving the students in this undergraduate project in an in-depth study on current software debugging methodologies and having the students apply their research findings to the development of a web-based debugging teaching tool for evaluation, analysis and tutoring. By participating in this undergraduate research project, 1) the students can gain exposure to the research of software quality assurance and acquire research expertise and experience in debugging techniques; 2) the students can gain hands-on experience in the design, develop and implementation of a complicated web-based application. The last objective of this project is to make the debugging teaching tool available to undergraduate computer science students.

Methodologies

One of the primary objectives of this project is to foster freshman and sophomore computer science students' interest in research and engaged learning. It is believed that this objective can only be achieved by having students actively engage in research

activities. The experimental nature of computer science also suggests that learning can be more effective if knowledge acquired is applied to the application development. Therefore, in this undergraduate project, the students are required to engage in both research and development activities.

The context of this undergraduate research project is the research field of debugging methodologies, particularly, teaching debugging skills to novice students. Four research activities have been conducted. The first research activity is expected to shed light on the following general debugging related questions: 1) what skills are important for debugging? 2) what types of bugs are there? 3) what are the causes of each type of bugs? 4) what are the most difficult tasks in debugging for novice programmers? 5) what debugging strategies are most effective? To answer those questions, the students are asked to gather relevant information on software quality assurance, particularly debugging methodologies and debugging teaching methods and tools, from research literature, books and the Internet. After identifying the key factors affecting debugging performance, the second research activity is to collect, select, categorize, and compile a database of questions that can be used to evaluate a student's strength and weakness in those key areas. This database provides the quizzes to assess a student's debugging skills. Each question is categorized according to skill types, bug types, bug source types and behavior types, etc. identified at the end of the first activity. The third research activity is to search, select and develop tutorial topics and practice exercises that cover those areas affecting debugging performance. Tutorial topics and practice exercises are also associated with skill types, bug types, bug source types and/or behavior types. The last research activity is to review current available teaching debugging tools, evaluate and compare the content and pedagogical approaches used. The results of the second, third and fourth activities build the necessary ground work for the development phase for this project.

The development phase of this undergraduate research project incorporates the previous research findings into the development of a web-based software tool for assessment, analysis, tutoring and practicing debugging skills. Here are more detailed features for this tool:

- The web-based tool creates debugging skill assessment quizzes. Quizzes are made of the randomly selected questions from the database produced at the end of the second research activity. Since questions in each quiz are categorized, the software can evaluate a student's areas of strength and weakness based on his/her assessment results.
- Since tutorial topics and practice exercises are also categorized, after identifying the skill level of a student, the application recommends a list of tutorials and practices to target at the areas to be improved.
- The web-based application keeps track of a student's activities such as his/her assessment results and tutorial and exercises completion status. Instructors can use this feature to monitor a student's progress and improvement.

Reach findings and development results

The in-depth literature research on debugging and teaching debugging methodologies conducted in the project reveals that: 1) The comprehension of domain knowledge, program and compiler message are key skills to better debugging performance [2, 15]; Recent research results also show that some cognitive skills such as meta-cognitive awareness also plays important role in debugging success [2]; 2) Locating bugs is more difficult for novice programmers than fixing them [2, 5]; 3) Bugs produced by freshman and sophomore students are mostly categorized using the categorization by Johnson, et al., [4]: a necessary operation or component is missing, a unnecessary operation or component is added, a necessary operation or component is misplaced, and a necessary operation or component is incorrectly implemented; 4) In general, bugs are caused by fragile knowledge, as summarized in [7, 11]. Fragile knowledge is divided into four types: knowledge that has not been acquired, knowledge that is unable to be retrieved, knowledge that is used in the wrong place and knowledge that is combined in the wrong way; 5) There is not best debugging strategies. An effective strategy helps programmers discover a bug quickly. Therefore guidance on when different strategies are used is more important and beneficial.

The research also results in a rich collection of quiz questions, tutorials and exercises on debugging strategies, programming constructs, and code and error message comprehension. Each quiz question, tutorial topic and exercise is associated with a bug type, a skill type, a knowledge type, and/or in some cases, a programming construct. Although most studies focus on non-syntactic errors, the data collected in [5] indicated that twenty percent of the bugs are caused by syntax errors. In reality, for novice programmers, syntax errors are just as time-consuming and frustrating to find as logic errors. Therefore, in this research project, syntax errors for C++, along with error messages generated by a number of popular compilers (e.g., GNU g++, Microsoft .NET and Borland C++), are also gathered and categorized.

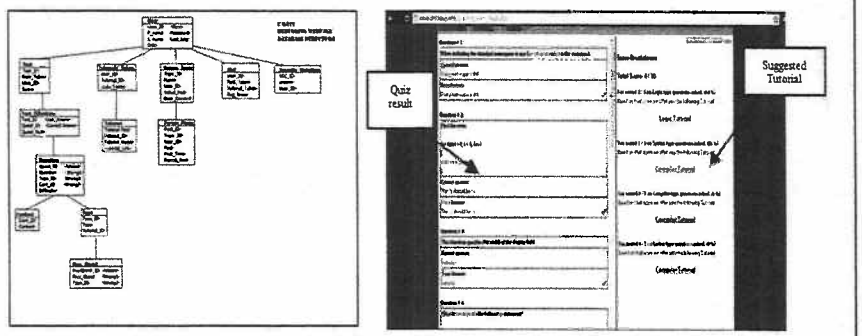

Figure 1. Database Diagram and Quiz Summary Screen

More than twenty debugging tutorial websites and tools are evaluated in this undergraduate research project. The content and pedagogical approaches among those websites and tools are reviewed and compared. The result shows that not many tools

provide the assessment of debugging skill levels and currently there are very little skill-based tutorials available. This motivates and drives the entire design and development of the web-based debugging tool in this project. The categorization of questions, tutorials and practices is very important for the implementation of this application because it relates a specific quiz question with appropriate tutorials and practice exercises. This relationship enables the web-based application to deliver skill-based material to each individual (Figure 1 shows the relationship between quiz result and tutorial content).

Another important function implemented in this debugging teaching supplement tool is to keep track of each student's activities such as his/her assessment results, and his/her progress in tutorial and training exercises. According to the research conducted by the students in this project, very little work has been done in systematically and quantitatively evaluating the effectiveness of debugging skill teaching approaches. This is mainly due to the limitation of time, cost and resources. Hopefully the data collected by this application can help provide some quantitative measurement of a student's progress and improvement, and provide quantitative evidence of the effectiveness of the debugging methodologies used.

This debugging skill assessment and tutoring tool is to be used as a teaching supplement for several lower level programming courses. The web-based architecture is adopted so that the application is easily accessible from anywhere. The front-end forms are written in PHP script language and backend database is supported by mySql. PHP and mySql are the two widely used tools for web-based applications. They are open source and offer many advantages. Currently they are not being used in the first and second year computer science courses yet. In this project, the students learn to use those tools while working on the application and gain tremendous knowledge and experience in developing a web-based application with PHP and mySql. In the end, not only do they acquire several desirable technical skills, they also gain a lot of experience in applied learning.

CONCLUSIONS

The duration of this undergraduate project is one year. Five undergraduate computer science students are involved in the research and development at different stages of the project. Over fifty undergraduate computer science students participate in the testing of the final product.

By participating the undergraduate research and development activities, the students are introduced to the research area of software quality assurance, enrich their knowledge and experience in debugging methodologies and debugging teaching approaches, and gain first-hand experience in applying their findings to the development of a skill-based debugging teaching tool.

ACKNOWLEDGEMENT

This project is funded through Bridge for SUCCESS undergraduate research grant from the generous support of the NSF's Science, Technology, Engineering, and Mathematics Talent Expansion Program (STEP).

REFERENCES

[1] Anewalt, K, 2005, Using peer review as a vehicle for communication skill development and active learning, *Journal of Computing Sciences in Colleges*, Vol. 21 , No. 2, pp. 148 - 155, December 2005.

[2] Fitzgerald, S. , G. Lewandowski, R. McCauley, L. Murphy, B. Simon, L. Thomas and C. Zander, 2008, Debugging: finding, fixing and failing, a multi-institutional study of novice debuggers, *Computer Science Education*, Vol. 18, No. 2, June 2008, 93-116.

[3] Gehringer, E.F., D.D. Chin, M.A. Perez-Quinones, and M.A. Ardis, 2005, Panel: Using peer review in teaching computing, *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, Vol. 37 No.1, February 2005.

[4] Johnson, L., Soloway E., Cutler, B., & Draper, S., *Bug Catalogue: I* (Technical Report No. 286), New Haven, CT: Yale University, Department of Computer Science.

[5] Katz, I., and J. Anderson, 1987, Debugging: An analysis of bug location strategies, *Human-Computer Interaction*, 3(4), 351-399.

[6] Kazemian, F. and T. Howles, 2008, Teaching challenges: testing and debugging skills for novice programmers, *Software Quality Professional*, Vol. 11, No. 1, 2008, 5-12.

[7] Ko, A., and B. Myers, 2005, A framework and methodology for studying the causes of software errors in programming systems, *Journal of Visual Languages and Computing*, 16, 41-84.

[8] Laporte, C.Y., A. April, and K. Bencherid, 2007, Teaching software quality assurance in an undergraduate software engineering program, *Software Quality Professional*, Vol. 9, No. 3, 2007, 4-10.

[9] Lu, H. and X. Wang, 2006, Learner-centered technical review in programming courses, *Proceedings of the International Conference on Software Engineering Research and Practice*, June 23-26, 2006, 702-709.

[10] McCauley, R. , S. Fitzgerald, G. Lewandowski, L. Murphy, B. Simon, L. Thomas and C. Zander, 2008, Debugging: a review of the literature from an educational perspective, *Computer Science Education*, Vol. 18, No. 2, June 2008, 67-92.

[11] Perkins, D., and F. Martin, 1986, Fragile knowledge and neglected strategies in novice programmers in E. Soloway & S. Iyengar (Eds.), *Empirical studies of programmer*, (pp. 213-229), Norwood, NJ: Ablex.

[12] Roberts, E., and G. Engel, eds, 2001, *Computing Curricula 2001: Final report of the joint ACM/IEEE-CS task force on computer science education*, Los Alamitos, California, IEEE Computer Society Press.

[13] Wang, X., Introducing Personal Software Process in a Small Computer Science Program Proceedings of the International Conference on Software Engineering Research and Practice, Las Vegas, Nevada, June 27-30, 2005, 731-737.

[14] Wang, X., Teaching Programming Skills through Learner-Centered Technical Reviews for Novice Programmers, *Software Quality Professional*, Vol. 13, No. 1, December 2010, 22-28.

[15] Zeller, A., 2009, *Why programs fail - a guide to systematic debugging*, 2nd ed., Morgan Kaufmann.