ISSN 2663-8169 Vol. 4, Issue 17: 5-12, DOI: 10.25236/IJNDE.2022.041702

# Debug Education: Students Debug Ability Assessment

**Fuyi Dong**

*College of Science, University of Auckland, Auckland, New Zealand*

***Abstract:*** *This paper reviews the relevant literature on computer program debugging by students. Debugging is a part of programming learning, the complete programming process will be the end of the program debugging, but this step for the novice programmer has a certain difficulty. Although there are many programming AIDS to help beginners quickly start programming, it is difficult to avoid mistakes in the programming process. Experts have a wealth of programming experience and spend a certain amount of time on debugging. For novice debuggers with a more unfamiliar programming environment, debugging will take more time. This paper reviews and summarizes the debugging research literature in the past, focusing on the following four aspects: What is a novice debugger? Common types of novice errors and student performance in debugging. Finally, we hope to promote the future development of computer education through the analysis of these debugging studies.*

***Keywords:*** *student debugging, debugging, education, program*

## 1. Introduction

We began to review the literature related to student debugging, trying to understand the problems students encounter when learning programming, as part of the research on computer teaching. When newbies are first introduced to programming, they are unfamiliar with the programming environment[7]. In the process of programming and learning in a strange environment, beginners will always make all kinds of mistakes, which will lead to the program can not normally achieve the expected results, and at this time, it is necessary to debug the program. According to Michaeli and Romeike(2020)[13], even experts spend 20 to 40 percent of their programming time debugging. Novice debugging takes much more time than expert debugging, and the novice is significantly less likely to correct errors. Even in the course of most debugging, newbies will introduce at least one new bug, making it harder for them to fix the bug[11]. Interestingly, some students have experience with programming. They may have some knowledge or even experience with programming languages such as JAVA. Students who have programming experience may perform differently in debugging than those who have no programming experience at all. This article includes an explanation of the definition of novice debugging, statistics of common error types for novice debugging and students' performance in debugging. Our purpose in writing.

This review is to contribute to the direction of computer education research by summarizing the different aspects of novice debugging.

We will focus on the issues that computer science teachers are more concerned about, especially those related to student debugging. We focused more on the performance of novice programmers in debugging. Although there were students who had programming experience, most of them had no direct exposure to formal debugging education, although the performance of experienced students was compared with that of complete novice programmers. In some cases, data on debugging for novice programmers will also be included, and the problems faced by novice programmers and novice students will be brought together for discussion.

Our survey of papers and literature focuses on the following four aspects:

•What is a novice debugger

•Common types of errors for novices

•Students' performance in debugging

We believe that through the study of these four aspects, we can more clearly show all aspects of student debugging, and more help and inspiration to computer education workers' debugging education in the future.

ISSN 2663-8169 Vol. 4, Issue 17: 5-12, DOI: 10.25236/IJNDE.2022.041702

This review discusses each of these four aspects in a different section. In Section 2 we will use the Dreyfus model to discuss the question "What is a debugger novice?" The Dreyfus model divides the different stages of the debugger, which can be combined with examples to give a grade to student debuggers.

In Section 3, "Common Error types for Novices", we reviewed a lot about the types of errors that novice debuggers often make in debugging experiments, as well as the types of errors that are common in code being debugged by novices.

In section 4 "Student's Performance in Debugging", we studied and analyzed several experiments about debugging, especially in section 3 about novice's performance in the process of fixing some common code errors. Among them, whether programming experience will affect students' performance in debugging will also be discussed in this section.

In the conclusion section, we will discuss the direction of further development and improvement of debugging teaching based on the existing research data, as well as the direction of future computer education research.

## 2. What is a novice debugger?

What crowd is called a novice debugger? This section uses the Dreyfus model to explain the different stages of debugging development. Further discussion is carried out by classifying groups with different debugging mastery and combining with the performance of students in debugging in reality.

Dreyfus model is a model that summarizes the performance of different stages of debugging, which can be divided into five stages: novice, advanced beginner, competent, proficient, and expert.As shown in Table 1, the research of Chmiel and Loui (2004) [6] is shown in combination with the explanation of the five stages of Dreyfus model, as well as Specific Performance applied to debugging.This model shows debugging in combination with the Dreyfus model. Debuggers can advance to the next stage by improving their debugging experience and skills. Chmiel and Loui's study also noted that although a student could identify simple errors, the student was still new at solving complex errors.

*Table 1: Modules on Debug Stage and Specific Performance by Chmiel and Loui (2004)*

| Stage | Explain | Specific Performance |
|---|---|---|
| novice | Not focusing on the whole.<br>Never challenge the rules but follow them completely.<br>Frustrated by an inability to handle errors. | • Lack of debugging methods<br>• Make the same mistake<br>• Unplanned debugging<br>• Spend a lot of time<br>• Rely on the help of others<br>• Easily lose heart and become depressed |
| advanced beginner | Future events will be based on the same experiences of the past.<br>Document past experiences and learn from them.<br>Never give up easily. | • Develop debugging skills through past experience<br>• Occasionally make the same mistakes<br>• Learn from past mistakes<br>• Try other debugging techniques<br>• Some degree of dependence on others |
| competent | Summarize past experience into rules.<br>Build models that help them make decisions. | • Try more debugging techniques<br>• Decide to use different debugging techniques<br>• If the current method fails, it will be replaced with another method<br>• Do not rely on others for most debugging |
| proficient | Pay more attention to the big picture.<br>Optimize decision model to accelerate decision making ability. | • Debugging becomes part of application development<br>• Use techniques from different domains to facilitate debugging<br>• Optimize decision making and debugging methods<br>• Help others instead of asking for help |
| expert | Have a lot of experience to judge things for yourself.<br>Don't rely on rules.<br>Deal with more complex situations and mentor others. | • Use rich experience for debugging<br>• Resolve responsible or unfamiliar errors more quickly<br>• Help others instead of asking for help |

Specific descriptions of student performance in debugging are available in some other relevant literature. A related study by Whalley et al. (2021) [15] shows that the student debuts more often, lacking expertise and a more systematic debug plan, by randomly trying out the different debugging methods he or she has learned. Try to find the error. Students lack a holistic view and often look for errors by repeatedly reading code line by line. They often rely on luck rather than analyzing the cause of the

ISSN 2663-8169 Vol. 4, Issue 17: 5-12, DOI: 10.25236/IJNDE.2022.041702

problem to locate the error, a process that sometimes takes a lot of time, which leads to inefficient debugging. These reasons cause many students to hold a negative attitude towards debugging. They think the debugging process is very tedious and takes a lot of time and energy, and even lose confidence in programming learning because of frustration.

Therefore, we can classify students according to their performance in debugging. Although some students with programming experience perform better on simple debugging tasks than students without any programming experience, they usually do not have formal debugging training. When students encounter problems, they will seek help, although the efficiency is not very good, and students often encounter the same mistakes in programming, and waste a lot of time on simple problems, so they begin to feel inferior and frustrated, and lose confidence in debugging[4]. To sum up, students can be classified as novice debuggers at this level.

## 3. Common type of novice errors

A large number of experiments on debugging research are specific to different types of errors. In addition, common programming errors of novices mentioned in different literatures and the proportion of errors in experiments are summarized in Table 2.

*Table 2: Error type collation module*

| Citation | Author | Year | Types of Problems | Occurences |
|---|---|---|---|---|
| [3] | Albrecht and Grabowski | 2020 | spelling mistake/typo | 49 |
| | | | undeclared variable | 41 |
| | | | missing semicolon | 38 |
| | | | invalid syntax | 29 |
| | | | conflicting/incompatible types | 18 |
| | | | missing/misplacement of braces | 17 |
| | | | missing parentheses for method call | 15 |
| | | | unhandled exception | 11 |
| | | | array used in place of array element | 6 |
| | | | variable needs a unique name | 6 |
| [5] | Chan Mow | 2012 | variable not found | 49.8 |
| | | | Identifier expected | 14 |
| | | | Class not found | 5 |
| | | | Mismatched brackets/parentheses | 5.3 |
| | | | Invalid method Declaration | 3.7 |
| | | | Illegal start of type | 3.4 |
| | | | Method not found | 1.6 |
| [1] | Ahmadzadeh et al. | 2005 | field Not found | 60.2 |
| | | | Use of Non-Static Variable Inside the Static Method | 21.8 |
| | | | Type Mismatch | 7 |
| | | | Using a non-Initialised Variable | 5.1 |
| | | | Method Call with Wrong Arguments | 3.9 |
| | | | Method Name Not Found | 1.9 |
| [4] | Alqadi and Maletic | 2017 | Using a single ' = ' when ' == ' is intended | 30 |
| | | | Loop has no body- extra semicolon | 55 |
| | | | Misusing Switch statement, missing break. | 29 |
| | | | Misusing && and \|\| operators | 14 |
| | | | Division by integers, so quotient gets truncated | 19 |
| | | | Condition variable has not been updated | 39 |
| | | | Array index out of bounds | 26 |
| | | | Attempt to access deleted dynamic variable | 9 |
| [11] | McCauley et al. | 2008 | Zero is excluded | |
| | | | Output fragment | |
| | | | Off-by-one | |
| | | | Wrong constant | |
| | | | Wrong formula | |

According to the summary and proportion of common novice errors in different experiments in Table

ISSN 2663-8169 Vol. 4, Issue 17: 5-12, DOI: 10.25236/IJNDE.2022.041702

3, these different types of errors can be classified and summarized. Among them, Syntax, Semantic and Logic are the most common, because students' understanding of programming itself is limited. Among them, Michaeli et al. (2019) [12] puts forward that Syntax is the most common error in programming. This view is also confirmed in the experimental research of Chan Mow (2012) [5].

### *3.1 Syntax Error*

Different debugging methods are used for different types of errors. Among them, Syntax is the most common error, accounting for almost 90 percent of students' mistakes, but it is not a very scary type of error. In fact, grammar errors are usually just spelling or punctuation errors that are easily picked up by compilers. Nowadays, many common compilers will automatically provide the error location and error information for the programmer when reporting an error, so that the programmer can locate the error location more easily and save the time to find the error. For students, being able to spot errors more quickly means having more time to think about them and correct them, and grammar errors are often easily corrected (Table 3).

*Table 3: Three Common Errors of Chan Mow (2012)*

| Error | Frequency |
|---|---|
| Syntax | 94.1 |
| Semantic | 4.7 |
| Logic | 1.2 |

### *3.2 Reasons for Mistakes*

Although the most common syntax error is not a very difficult error, the high frequency of this error will lead to a long time for debugging. Why students often make mistakes in programming is also worth studying. In 2020, Albrecht and Grabowski [3] collected and sorted out the common errors students make in programming, and also sorted out some of the most common reasons why students make mistakes. The reasons for different degrees of programming errors are also different, among which the most common reasons are summarized as: Syntact, Concept, Strat, sloppiness, misinterpret, domain. Of these reasons, sloppiness and misinterpret topped the list for making mistakes at 34 percent and 24 percent, respectively. In particular, these two reasons account for the highest proportion of wrong Output format mistakes most frequently made by students. This shows that beginners often make more mistakes not because of the complex programming language or logic, but because they are not familiar with the programming environment and their own carelessness.

Through the summary and analysis of the relevant research on “common types of novice errors”, we get some enlightenment for debugging teaching. In the debugging teaching, teachers may need to emphasize the grammar problems of some programming languages and even remind students to check their spelling in the process of programming, which can improve the debugging efficiency more efficiently.

## 4. Students' performance in debugging

For experienced program developers, debugging as a part of the development program, they have rich debugging skills to quickly complete debugging. But students, as novices, do not have debugging strategies, so professional debugging tips and advice may not be appropriate for novice debuggers.

### *4.1 Debugging technique*

First, we need to understand the first debugging techniques a novice will use in different situations. Simon et al. (2008) [14] There is a statistical experiment on debugging knowledge, in the form of simulation test in order to explore the debugging skills that students have before receiving debugging learning. The main tests are in four areas: troubleshooting, fault locating, exploring unfamiliar environments, and describing the debugging experience in the real world.

Based on the data in Table 4, it can be found that novices use more abundant methods for troubleshooting, including providing information for repair operations, trying tests, proposing diagnosis,

ISSN 2663-8169 Vol. 4, Issue 17: 5-12, DOI: 10.25236/IJNDE.2022.041702

repeatedly repairing, using alternative solutions and even asking for help from others.For error location, the vast majority of people are testing related operations or diagnostics. When testing in an unfamiliar environment, more people turn to others for help. Testing and diagnosis are also two of the most common ways to solve problems in the real world. Therefore, when students are not familiar with the programming environment, they may choose to ask for help when they encounter program errors, so teachers, classmates and the Internet will become the objects for their help.

*Table 4: Student Debug Method Model from Simon et al. (2008)*

| Method | Troubleshooting | Locating | Unfamiliar | Real-world |
|---|---|---|---|---|
| Test operation | 38 | 71 | 15 | 46 |
| Provide information to fix | 52 | 20 | 50 | 83 |
| Other tests | 62 | 93 | 56 | 94 |
| Diagnose | 58 | 75 | - | 97 |
| Fix again | 64 | 21 | 59 | 60 |
| Backup | 61 | - | 24 | - |
| Ask for help | 51 | - | 82 | 34 |

### *4.2 Students debugging*

Next comes the part where students debug. Debugging is divided into four steps: understand the program, test the program, locate the error, and repair the error. For student debugging, correctly locating the error location and correctly repairing the error is called a successful debugging. Fitzgerald et al. (2008)[7] in an experimental study on student debugging, many students were invited to debug some problematic programs. The data of different problem types and student repair success rate are shown in Table 5. Apparently, more than half of the students were able to find and fix bugs in their programs, and most of the bugs that were fixed were due to programming language errors. Errors that do not involve programming languages, such as for loops or if statements, and initialization errors, are usually not easy to fix. However, for some non-structure-related errors, such as off-by-one loops or incorrect calculations leading to logical algorithms, it is difficult for students to debug. This is because at the error location step, the compiler can help locate structural errors, so they are easier to find and fix, whereas non-structural errors are harder to find and therefore harder to fix.

For the student debugging process, understanding someone else's code is much more difficult than debugging your own[15]. The overall understanding of the code and the functional requirements can make the debugging process easier, which requires a debugging strategy. A good strategy plays an important role in debugging, and professional developers are very efficient in choosing a debug strategy[12]. The logical debugging process is difficult for students. Most of the time, students just use print statements to repeatedly test the output of the code, so as to find the location of errors. However, students' choice of debugging strategies is very limited. They lack systematic debugging strategies and will not use appropriate debugging methods. Even more than half of students use less successful debugging methods, which further reduces the debugging efficiency. Therefore, instead of just explaining the errors encountered in the code, the more important direction of debugging education for teachers is the cultivation and practice of debugging strategies, so that students can develop a good self-debugging model, so that they can deal with more error debugging.

*Table 5: Student Debug Method Model from Simon et al. (2008)*

| Error Types | Ratio of Student Fixed |
|---|---|
| Language knowledge error | 86 |
| loop condition update | 71 |
| assignment | 100 |
| if body not compound | 71 |
| should not be inloop | 80 |
| conditional:incorrect if logic | 50 |
| loop:boolean logic | 75 |
| loop:iteration condition error | 50 |
| loop:nested loop variable swap | 75 |
| arithmetic error | 54 |
| data casting error | 67 |
| update error | 57 |

ISSN 2663-8169 Vol. 4, Issue 17: 5-12, DOI: 10.25236/IJNDE.2022.041702

### *4.3 Programming experience and debugging ability*

In this part, we will study whether students with programming experience in advance will be helpful for debugging ability from the introduction.

According to Katz and Anderson (1987)[9], the data in Table 6 are sorted out. It can be seen intuitively that people with more programming experience make fewer programming mistakes than people with less experience, and these programming experiences can also help them find the location of the error more accurately, thus assisting in debugging. Rich programming experience can help programmers better read and understand the overall content of the code, and can quickly locate errors according to the location, so as to facilitate subsequent repair. More programming experience also means more debugging experience. Experienced debuggers will accumulate and summarize mistakes made in the past, and when they encounter the same mistakes again, they can use their past memory and experience to debug.

However, the experimental results of Ahmadzadeh et al. (2005) [1] give somewhat different conclusions. The people who took part in this experiment were students, some of them were good at programming and some of them were bad at programming. In this experiment, only a few students completed all of the final debugging, and most did not, although a few of them were able to fix some but not all of the errors. The final conclusion of the experiment is that those students with excellent debugging ability have rich programming experience and excellent grades. But not all students with good programming scores are good debuggers. Students who were good at programming but weak at debugging often had problems locating errors, and some isolated errors rather than correcting them.

To sum up, combined with the content in Section 2, even if students have some programming experience, they are still beginners in programming. Rich programming experience for debugging is a lot of help, can help programmers more efficient debugging. It is still far from enough for students to master programming experience, and they still need help in debugging. Therefore, computer teachers debugging teaching guidance, for students is still very necessary.

*Table 6: Programming and debugging with different programming experience from Katz and Anderson (1987)*

| Test | Program error | Error detection |
|---|---|---|
| Rich programming experience | 46.4 | 71.4 |
| Medium programming experience | 68.1 | 55.6 |
| Less programming experience | 77.3 | 53.1 |

## 5. Discussion

In this section we will combine the three aspects of the first section to discuss our findings and discuss some of the implications for computer teaching. What is a debugger? According to the Dreyfus model combined with the actual situation of debugging analysis [6], for students who are new to programming or have already had programming experience, there is still very little useful programming experience in nature. Students have not received formal programming training before, and debugging, as a part of programming, is highly dependent on programming experience, so students can be uniformly regarded as debugging novices. Students will react like novice debuggers. Therefore, they also need help from others to master basic debugging skills as soon as possible, and gradually develop their own debugging skills and strategies.

Common types of novice errors although many studies have focused on Logic errors that are harder to find and more complex, our study focuses more on Syntax problems. While the Syntax problem is fairly simple for programmers, it does occur the most frequently. For students, diving straight into more complex debugging can be tedious and time-consuming and lead to loss of confidence [15]. Starting from simple debugging can help them build programming confidence, but also can remind students to pay attention to their careless mistakes, urge students to pay more attention to the most common details in programming, so as to reduce the frequency of subsequent programming mistakes, but also improve debugging efficiency.

Student performance in debugging in this section, we study student debugging from three aspects. The first aspect is that when students have no contact with programming, some of their own real life experience will spontaneously conduct debugging when facing problems. And by understanding these

ISSN 2663-8169 Vol. 4, Issue 17: 5-12, DOI: 10.25236/IJNDE.2022.041702

students spontaneously with debugging skills, can better help teachers according to after debugging skills to guide and improve, train the students' ability to propose multiple hypotheses, can effectively solve the problem, also can cultivate students' independent ability to solve the debug [13]. The first aspect is the specific study of student debugging. It can be found that some specific problems of students in debugging and the use of wrong debugging skills lead to low debugging efficiency. Some of these students can write code normally but cannot modify the same type of code in debugging. Some students don't understand the program and don't know what to do. Others lack a debugging strategy, and when they encounter multiple problems, they will consider whether the previously fixed bug failed rather than looking for other bugs[10]. Students lack some debugging experience as the foundation, so it can be in the daily practice of debugging, encourage students to summarize their experience, and get inspired for the debug mode that suits oneself [13].

We give some suggestions for teachers in debugging teaching. In the past, debugging teaching focused more on correcting the problems encountered by teachers in the programming process. Our study focuses solely on the performance of students during the formal debugging training. Debugging needs more diversified contact activities rather than one or two targeted activities, master more diversified debugging skills can better improve students' debugging ability[6]. On the one hand, when teaching programming courses, computer teachers should teach the whole programming process, that is, designing, writing and debugging, instead of just teaching the writing of code[7]. Although the teaching of coding can help students understand and fix Syntax problems, the complete programming process can help students better read others' codes, understand the code design ideas, and fix non-SYNTAX class problems in the code. On the other hand, debugging errors encountered by students in programming are often repeated, so students can share with each other the process of encountering errors and solving them, and also accumulate debugging experience together[8]. Moreover, good debugging practice can significantly reduce students' time spent on programming assignments and increase their confidence in programming. Debugging is a complex skill, but it is so important in programming that it takes practice to master. The addition of debugging related courses can effectively help students to increase the number and time of debugging practice and speed up their grasp of debugging skills and debugging strategies.

## 6. Threats to validity

There are more aspects to consider when evaluating these results. More notably, compared with previous studies, our study focused on the most common problems encountered by novice debuggers and did not focus on the more complex problems. For novice debuggers, especially students, the most common errors require more attention, because the reduction of these errors greatly improves the efficiency of debugging and allows more time and energy to focus on other, more complex problems. We believe that only by laying a good foundation for debugging and letting students relax and learn debugging with confidence, the subsequent debugging teaching will have a better effect, rather than focusing on complex debugging problems at the beginning.

Although the subsequent debugging teaching will still encounter those complicated debugging problems, when the students are confident that they can fix the problem, whether they seek help or explore by themselves will be an active process, rather than passively accepting debugging skills. While students are actively learning debugging skills, they later reflect on their programming problems, which helps them gain experience and knowledge from their failures and make for a better debugger in the future[15]. Programming ability and debugging are mutually reinforcing, and good debugging is part of programming[2]. Establish a good debugging mentality from the beginning, students are more likely to accept debugging teaching, so as to improve their programming ability.

## 7. Conclusion

Through the analysis and summary of the past experimental research literature, it is confirmed that some of the results are still valid for today's novice teaching. As a computer science teacher, the best hope of course is that when a student has enough expertise to find the error, that expertise is likely to be enough to fix it[8]. But for students, sufficient professional programming knowledge still needs to be gradually accumulated and mastered through teaching and practice.

Compared with previous studies, the findings of this study lay more emphasis on gradually cultivating students' adjustment and progress, especially in the aspects of interest and confidence. Chmiel and Loui(2004)[6] once proposed that the best debugging teaching is not focused on one or two aspects of

ISSN 2663-8169 Vol. 4, Issue 17: 5-12, DOI: 10.25236/IJNDE.2022.041702

activities, but diversified teaching, students can use a variety of debugging skills to think about which skills can solve the problem in front of them faster. This thinking process will accelerate the master of debugging skills and gradually cultivate students' debugging strategy model, helping them to enter more advanced debugging stage faster. The results of this study provide a direction for the future computer teaching work, and hope that the future computer teaching workers in guiding students to learn programming spontaneously and actively, and can gradually develop good programming habits to make some contributions.

## References

*[1] Marzieh Ahmadzadeh, Dave Elliman, and Colin Higgins. 2005. An analysis of patterns of debugging among novice computer science students. ACM SIGCSE Bulletin 37, 3 (9 2005), p.84-88.*

*[2] Marzieh Ahmadzadeh, Dave Elliman, and Colin Higgins. 2007. The impact of improving debugging skill on programming ability. Innovation in Teaching and Learning in Information and Computer Sciences 6, 4 (10 2007), p.72-87.*

*[3] Ella Albrecht and Jens Grabowski. 2020. Sometimes it's just sloppiness Studying students' programming errors and misconceptions. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education. ACM, New York, NY, USA.*

*[4] Basma S. Alqadi and Jonathan I. Maletic. 2017. An empirical study of debugging patterns among novices programmers. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education. ACM, New York, NY, USA.*

*[5] Ioana Chan Mow. 2012. Analyses of Student Programming Errors In Java Programming Courses. Analyses of Student Programming Errors In Java Programming Courses Journal of Emerging Trends in Computing and Information Sciences 3, 5 (2012), p.740-749.*

*[6] Ryan Chmiel and Michael C. Loui. 2004. Debugging: From Novice to Expert. ACM SIGCSE Bulletin 36, 1 (3 2004), p.17-21.*

*[7] Sue Fitzgerald, Gary Lewandowski, Renée McCauley, Laurie Murphy, Beth Simon, Lynda Thomas, and Carol Zander. 2008. Debugging: Finding, fixing and flailing, a multi-institutional study of novice debuggers. Computer Science Education 18, 2 (6 2008), p.93-116.*

*[8] Yasmin B. Kafai, David DeLiema, Deborah A. Fields, Gary Lewandowski, and Colleen Lewis. 2019. Rethinking debugging as productive failure for CS education. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education. ACM, New York, NY, USA.*

*[9] Irvin Katz and John Anderson. 1987. Debugging: An analysis of bug-location strategies. Human-Computer Interaction 3, 4 (dec 1 1987), p.351-399.*

*[10] Zhongxiu Liu, Rui Zhi, Andrew Hicks, and Tiffany Barnes. 2017. Understanding problem solving behavior of 6-8 graders in a debugging game. Computer Science Education 27, 1 (jan 2 2017), p.1-29.*

*[11] Renée McCauley, Sue Fitzgerald, Gary Lewandowski, Laurie Murphy, Beth Simon, Lynda Thomas, and Carol Zander. 2008. Debugging: A review of the literature from an educational perspective. Computer Science Education 18, 2 (6 2008), p.67-92.*

*[12] Tilman Michaeli and Ralf Romeike. 2019. Improving Debugging Skills in the Classroom - The Effects of Teaching a Systematic Debugging Process. In Proceedings of the 14th Workshop in Primary and Secondary Computing Education. ACM, New York, NY, USA.*

*[13] Tilman Michaeli and Ralf Romeike. 2020. Investigating students' preexisting debugging traits: A real world escape room study. In Koli Calling '20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research. ACM, New York, NY, USA.*

*[14] Beth Simon, Dennis Bouvier, Tzu-Yi Chen, Gary Lewandowski, Robert McCartney, and Kate Sanders. 2008. Common sense computing (episode 4): Debugging. Computer Science Education 18, 2 (6 2008), p.117-133.*

*[15] Jacqueline Whalley, Amber Settle, and Andrew Luxton-Reilly. 2021. Novice reflections on debugging. In Proceedings of the 52nd ACM Technical Symposium on Computer Science Education. ACM, New York, NY, USA.*