Thinking Strategically, Acting Tactically: The Emotions Behind the Cognitive Process of Debugging in Early Childhood

A thesis submitted by

Megan A. Bennie

in partial fulfillment of the requirements for the degree of

Master of Arts

in

Child Study and Human Development

Tufts University

May 2020

Abstract

Debugging is an essential component of computer science. Previous research on coding in high school students, university students, and professionals over the past few decades has analyzed the cognitive process of debugging. Debugging requires an elaborate repertoire of strategies and tactics to approach a range in variety of errors. However, there is little to no research on the strategical and tactical elements of debugging in early childhood coders. Debugging skills are used at all levels of coding from early coders to advanced, but there are gaps in identifying what those strategic thought processes look like in early childhood. This research investigates the debugging process of children 5-7 years old using the KIBO robotics kit. The KIBO robotics kit is a tangible coding interface that uses wooden programming blocks. A mixed method approach focuses on identifying the tactics and strategies displayed while children debug errors and the role emotions play in this process. Results show a wide variety of strategies and tactics that early childhood coders use when debugging as well as themes across participants. Implications of this research will further the literature on debugging in early childhood.

Keywords: debugging, early childhood, problem solving, strategies, tactics, emotions

Acknowledgements

I would like to acknowledge the people who supported me throughout my graduate studies and the writing of my thesis. First, I must acknowledge my committee members, all of which are exceptional academics. Without the feedback, support, and guidance of each of my committee members, my mentor and thesis advisor Dr. Marina Umaschi Bers, Dr. Martha Pott, and Dr. Jennifer Cross, my research would not have been possible.

I am also truly thankful for the consistent support of my family and friends, the Eliot Pearson M.A. graduating class of 2020, and all of the DevTech lab members. Each of the aforementioned individuals provided me with encouragement and motivation along the way. This would not have been possible without the dedicated time, love, and moral support I received through and through.

DevTech Research lab provided an outlet for my exploration into research. I would be remised to call out the support of all the wonderful graduate students and undergraduate students of the DevTech Research lab. I am grateful for each of the families and children who participated in my research, and those who made conducting my research possible.

Most of all, I am grateful for the support of my family and Ryan, for the unconditional love they provided me with through all of the ups and downs throughout this process.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Problem-solving is an essential skill used in everyday life and contributes to the functioning of our society (Denham & Bouril, 1994; Klahr & Robinson, 1981; H. A. Simon & Newell, 1971). People engage in problem-solving at work, home, and with their friends. Problem-solving can be seen in an academic context in regard to math and memory tasks (Harris & Saarni, 1991). Likewise, problem solving is a fundamental skill necessary to debug errors in computer programs. In relation to computer programming, the term "debugging" is the process of correcting erroneous code, which in itself is the act of problem-solving (O'Dell, 2017). Debugging a problem is a process that requires several steps and *strategies* to fully comprehend the error and resolve it (McCauley et al., 2008).

Debugging, and the cognitive process it requires, are areas that have been thoroughly researched in middle school students, high school students, university students, and professionals with regards to the approaches used to solve errors. The act of debugging happens during all stages of coding regardless of skill level or programming knowledge (Falahah et al., 2015). In addition, debugging is described as challenging or even frustrating and emotions play a vital role as moods and emotions can make it difficult to perform on a task (Ekman, 2003). Children's cognitive function is influenced by emotion; poorly controlled emotions and negative emotions inhibit cognitive ability, as opposed to positive emotions and well-regulated emotions which enhances cognitive functioning (Raver, Garner, & Smith-Donald, 2007). This process was explored in the literature, with less emphasis towards an early childhood perspective. While it is known that young children debug while coding, less is known about the approach and processes employed. Based on previous literature on the debugging process, and its associated strategies and tactics, the information is not as easily accessible with early childhood coders because they're not aware of the specific named strategies and tactics that are explicitly taught in programming courses or referenced on on-line platforms.

This thesis will explore the following research questions:

1. What kind of strategies and tactics do early childhood coders use when debugging?
2. What role does emotion play during the process of debugging in early childhood?

The goal is to provide an in-depth analysis of the debugging process used in early childhood, specifically when children debug errors using the KIBO robotics platform. Concurrently, the role of emotion will be examined throughout the process.

Chapter 2: Literature Review

The literature available for early childhood exploration of technology has substantially grown throughout the past decades. Research including the theoretical foundations of computational thinking skills and coding will be the first topic discussed. A thorough review of the available literature was conducted on debugging, the cognitive process of locating and fixing errors in computer code, and the strategies and tactics employed. However, there is little to no research on the debugging strategies and tactics used by early childhood coders. It is supplemented with research on the general debugging process observed in middle and high school student, university students, and professionals as a framework for how they debug errors in computer programs. This leads into further examination of four error types: syntactic, semantic, systemic, and correspondence. Systemic and correspondence errors are more suited to the KIBO robotics platform.

To address the problem-solving process in early childhood in coding, it is essential to zoom out, taking a broader view of what may be occurring when children debug code. Children start school and are expected to adjust to classroom norms, that they may or may not have previously been exposed to at home. During this developmental stage, children begin to understand their emotions and comprehend heightened feelings brought on by situations. As children learn to regulate their emotions, it is important to understand the role of cognition, decision making, and motivation on any given task.

Coding in Early Childhood

In the modern era of technology, children are surrounded by a variety of technologies. While learning to code using these technologies, children are engaged in a higher level of thinking, called computational thinking. Computational thinking is considered a broad set of thought processes that are used to problem solve and think in more abstract ways (McCauley et al., 2008). This type of thought process was considered primarily for computer scientists. However, Wing describes that this process of thinking is not limited to just computer scientists, but should be a universal and essential skill for everyone (Wing, 2006). The addition of programming and using concepts from computational thinking is an approach of the twenty-first century. Seymour Papert, in his book *Mindstorms: Children, Computers, and Powerful Ideas*, is credited for coining the term computational thinking, which was later elaborated on by Jeanette Wing (Papert, 1980).

The topic of computational thinking in an educational setting has primarily been seen in introductory computer science or programming courses. However, as identified by Buitrago Flórez (2017), teaching programming is the most effective approach for teaching computational thinking. Computational skills can benefit all types of students especially engaging with the development of advanced problem-solving and debugging skills (Lockwood & Mooney, 2017).

Through the ideas of computational thinking seven powerful ideas that are developmentally appropriate for young children were identified: algorithms, representation, hardware/software, debugging, design process, modularity, and control structures that are developmentally appropriate for early childhood (Bers, 2018). Computational thinking can provide young children with a set of tools to enhance their thinking abilities. The process combines multiple layers of abstraction in the cognitive process in a similar aspect as to how computer programmers develop algorithms (Buitrago Flórez et al., 2017).

The cognitive programming process is divided into several iterative stages. Each stage is cyclical and provides the opportunity to formulate a problem, generate a plan, code, debug errors in the code, and verification or validation (Gould & Drongowski, 1974). While early childhood coders are exploring coding platforms and interfaces, they're creating projects and telling stories. During this process, they're actively finding and fixing issues within the programs. The program may not be displaying or interpreting the codes the child creates and when the program doesn't do what they want it to they, go back to fix it. This process of managing frustration and finding a solution allows children to develop strategies for debugging their creations (Bers, 2018).

Debugging

Debugging is a complex cognitive process in programming that involves searching for, identifying, and locating an error in the program to remove or correct it, for the program to behave as intended (Araki et al., 1991; Falahah et al., 2015; Gough et al., 1994; Lauesen, 1979; Shaochun Xu & Rajlich, 2004; Yen et al., 2012). A simpler definition provided by (O'Dell, 2017) is "a domain specific term for problem-solving." While debugging is a central part of computer programming it can often be seen as an overwhelming or frustrating task that brings up many emotions (Alqadi & Maletic, 2017; Badiozamany & Wang, 2010; Sue Fitzgerald et al., 2008; Gough et al., 1994; Gugerty & Olson, 1986; Jeffries, 1982; Katz & Anderson, 1987; Kessler & Anderson, 1986; O'Dell, 2017; Perkins et al., 1986; Perkins & Martin, 1986).

Debugging is a difficult concept to learn and master, but is an essential skill to possess for computer programming (Badiozamany & Wang, 2010; Chen et al., 2017). The required actions used in debugging can be observed at all levels of programming from novice to expert (Falahah et al., 2015). The ability to debug code improves with practice and experience rather than explicitly being taught the skill (Badiozamany & Wang, 2010). The more practice and exposure to different types of errors prepares the coder for a multitude of debugging situations. With constant exposure, the coder gains confidence in their coding ability as they begin to conceptualize more complex debugging tasks (Ahmadzadeh et al., 2005). Even as programmers become more skilled and experienced, the programs they write become more complex requiring a considerable amount of time devoted to debugging errors (Jeffries, 1982). Xu compares computer programmers to medical doctors in a way such that a doctor is displayed with symptoms, a programmer is displayed with errors in code (error messages, wrong outputs, etc.) both have to diagnose the issue (locate the error) and provide a solution (repair the error). Therefore, as previously mentioned, the debugging of errors is a demanding and extensive cognitive process (Shaochun Xu & Rajlich, 2004).

Debugging Process

As debugging can be a demanding cognitive process, it has been reported to take three times longer than the actual coding of the program itself (Rubey, 1986). As programmers become more advanced with their coding skills, they write more complex programs, but are still spending a portion of time finding errors (Jeffries, 1982).

Vessey (1982), and Gould (1974), researched the cognitive process of debugging and characterized the activity as in iterative process of synthesizing, testing, and refining hypotheses about bug locations and repairs. A bug in the context of computer programming is an error within the program that causes the behavior of the program to be inconsistent with the expectations of the programmer or the user (Shaochun Xu & Rajlich, 2004)

There are several models, theories, processes, and goals observed within the debugging process. For example, troubleshooting an error by conducting a systemic search to find the cause of the error as it relates to overall system, with the goal of removing the error. Katz and Anderson (1987), defined four stages of debugging, 1) understand the system, 2) test the system, 3) locate the error, and 4) repair the error. Katz and Anderson (1987) found strong evidence for each stage for debugging with each being separate, and each requires a particular set of skills. In that vein, the techniques needed to understand the system (stage 1), may not necessarily be contingent on or similar to the techniques needed to locate the error (stage 3). Understanding the system is less applicable to programmers, who are responsible for debugging their own code. A byproduct of writing your own code is you gain an implicit understanding of the perceived functionality and purpose of the program. However, for the programmers who debug others programmers code, than the first stage of understanding the system is much more essential (S. Fitzgerald et al., 2010; Katz & Anderson, 1987). The most difficult stage of troubleshooting for subjects while debugging is finding the location of the error (S. Fitzgerald et al., 2010).

The gross descriptive model of debugging is the process of how tactics are chosen and used in the process of debugging a program (Gould & Drongowski, 1974). This model entails the programmer selecting a particular debugging tactic, finding a clue to a bug (hypothesis), reporting the line containing the clue of error, and if nothing is detected then choosing another tactic. It is important for programmers to have a variety of tactics available for this model if not they may reach a “dead-end” without having another tactic to test.(Gould, 1975; Gould & Drongowski, 1974).

Learning to program is an extensive process and requires the use of computational thinking skills (Grover et al., 2014). As children continue to code, they gain more skills and knowledge about the programming processes. The real factors on the performance level of programmer's knowledge and skills are through their “bag of tricks” which are filled through substantial experience (Wiedenbeck, 1985). Debuggers are known to have a “bag of tricks” as to how they approach problems based on what they have previously seen.

At the introductory stages of coding, it can be overwhelming to an individual learning a new concept. It is often seen that programmers, who are struggling with code or an error message will turn to an instructor or more commonly the internet (B. Simon et al., 2007). However, most of the internet sources available require a level of program understanding that a novice level programmer may not have. This most often leads to frustrated students who deduce that programming is too difficult and confusing. Teaching and instilling the ideas of debugging as a process to work through at the beginning stages projects higher attrition rates within the computer science field (Kazemian & Howles, 2008). Harboring in on these “stuck” and difficult moments can motivate and channel the programmer to learn how to debug (B. Simon et al., 2007). Working through the errors and bugs of a program provides a coder with more skill and confidence which in time will excel them to the expert level of coding as they engage in writing more complex code and solving more complex bugs (Jeffries, 1982).

Expert coders, who have a great deal of programming knowledge are considered to be the individuals that build a “bag of tricks” for coding and approaching problems (Badiozamany & Wang, 2010). Greater experience and exposure to coding interfaces allows for more tricks and skills to be developed. These coders on average appear to understand programs better because of their exposure and advanced ability to comprehend problems which could lead to advanced debugging skills (Gugerty & Olson, 1986). A variety of literature, that compares expert and novice coders by the amount of time they have spent programming, it was found that the ability to chunk programs and therefore debug programs improve. However, literature does not support the idea that your coding identity as either expert or novice determines you debugging ability (Ahmadzadeh et al., 2005; Sue Fitzgerald et al., 2008; Kessler & Anderson, 1986; Murphy et al., 2008).

A study analyzing good programmers and the differences between their debugging skills proved that the majority of expert debuggers are also expert programmers, but less than half of expert programmers are expert debuggers (Badiozamany & Wang, 2010). These results provide a basis even though someone may portray high level coding skills that does not entirely transfer into having high level debugging skills. Through the investigation of the weak debuggers, Ahmadzadeh (2005) revealed that the "good programmer" with weak debugging skills would display knowledge of a debugging technique, but not the application technique. However, the considered "weak programmers" could not even locate the bug in the program. Overall, this provides the idea that the knowledge of application of debugging techniques and implementation of them into the program are key factors in limiting "good programmers" from being classified as good debuggers (Ahmadzadeh et al., 2005).

Even though programmers develop debugging skills through the experience of coding through them, it is challenging to learn independently (Klahr & Carver, 1988). Introductory programming courses encompass concepts of algorithms and data structures specifically instructing on essential programming concepts including strategies and tactics to use while debugging (Robert Charles Metzger, 2004).

Strategies & Tactics

Chinese military strategist, Sun Tzu in his book *The Art of War* wrote: "Strategy without Tactics is the slowest route to victory. Tactics without strategy is the noise before defeat." Exemplifying that the two concepts, strategies and tactics, work together in achieving a goal. In the context of debugging code, a strategy is operationally defined here as a high-level, reasoned plan, to achieve the goal of identifying the error (Grigoreanu et al., 2009; Romero et al., 2007). Tactics are the low-level actions or debugging procedures used in identifying that error (Gould, 1975; Romero et al., 2007). The 1980's was the most productive decade for producing research and literature on debugging (McCauley et al., 2008). Gould (1975), began the process of analyzing debugging strategies and discovered that programmers used a variety of tactics in their debugging approach of choosing a strategy to find the buggy area within the code, generate a hypothesis and select a tactic until the error was located.

While observing strategies employed by students, Fitzgerald (2005) examined the strategies used by students in their first or second computer science course and identified that all students used a range of strategies, students used multiple strategies across each problem, students applied different strategies to a variety of questions, and students often used good strategies poorly. Overall, the success was not determined by which strategy the student chose, but rather the way they used the strategy (Sue Fitzgerald et al., 2005). Strategies are diverse and there is not a "one-size-fits-all" approach to debugging or learning programming concepts (McCartney et al., 2007). Not every strategy or tactic may work for every programmer, however that does not mean that specific strategies or tactics are poor while others are superior. Strategies and tactics have to work within the programmer's pre-existing factors such as knowledge or should meet their needs as they engage in the process of debugging (Gould & Drongowski, 1974).

There is potential for improvement considering debugging expertise, regardless of experience level. To improve debugging skills, studies have shown that programming should include the practice of reading others' code and any commentary left by the original author (Badiozamany & Wang, 2010). When a programmer attempts to debug an error, they generally have the knowledge of what tools they're using, and what they intend on doing with them. Those programmers do not exhibit the same skill set needed in order to debug an error from a program they did not produce. To debug others' code there is a period of time where the individual has to slow down to comprehend what the code does or what function it is intended to perform (Gugerty & Olson, 1986). This is a common type of debugging tactic, the act of slowing down and engaging in immersive comprehension when evaluating others' work (Ahmadzadeh et al., 2005; S. Fitzgerald et al., 2010; Gugerty & Olson, 1986; Weiser, 1982; Wiedenbeck, 1985). Katz and Anderson (1987), found that programmers took longer to debug others' code than their own because they spent additional time trying to comprehend the codebase before proceeding to compile, execute, or otherwise provoke or debug any errors.

Generally speaking, students tend to debug erroneous code using a backwards reasoning strategy (S. Fitzgerald et al., 2010; Gould & Drongowski, 1974). However, when debugging others' work, they use a forward reasoning strategy, this promotes greater comprehension of the program (S. Fitzgerald et al., 2010; Gould & Drongowski, 1974). There is a deluge of literature that explores debugging strategies and tactics based on the type of program, type of error, prior knowledge and experience level of the programmer or original author of the code (Ahmadzadeh et al., 2005; Romero et al., 2007).

Expert coders were more likely to take a breadth-first strategy by trying to understand the program. In contrast, novice coders tend to take a depth-first strategy by focusing on finding and fixing the error with little regard for the root cause of the error or overarching problem (S. Fitzgerald et al., 2010; Murphy et al., 2008; Vessey, 1985). Subsequently, experts tend to prefer a breadth-first strategy, spending adequate time to fully comprehend the program and problem space before approaching the error. The difference in approach across experience level, suggests that novice programmers are at a disadvantage, as they may not have the knowledge or skills to adequately comprehend the program before independently exploring or attempting to debug an error, which can be quite time consuming.

Some programmers believe that working backward can be an effective debugging strategy (Badiozamany & Wang, 2010). Working backwards involves finding where the program starts to behave unexpectedly by tracing back from the location or line of code the error originates from (Badiozamany & Wang, 2010). In this example, the programmer examines the program thoroughly by utilizing a tactic called "Tracing". Tracing is classified as the act of methodically stepping through a programs code, going line-by-line, to locate an error.

Ducassé & Emde (1988), identified four "global" debugging strategies, which include, "filtering", "checking computational equivalence", "checking well-formedness", "recognizing stereotyped errors", each with underlying tactics to describe the utility of each strategy in effectively examining code to locate errors.

1. Filtering: Is the act of reducing the amount of information within a program to localize a bug. For example, common recipes when employing this strategy include:
   a. Tracing algorithms
   b. Tracing scenarios
   c. Path rules
   d. Slicing/dicing

2. Checking computational equivalence: This is very hard to achieve in the context of real-world programming as it necessitates knowledge of the intended program, or a preconceived understanding of the program, along with the program in question. For example, common recipes when employing this strategy include:
    a. Algorithm recognition
    b. Program transformation
    c. Assertions
3. Checking well-formedness: No prior knowledge of the intended program is required for this strategy. However, knowledge of the programming language, along with general programming knowledge is required. For example, common recipes when employing this strategy include:
    a. Language consistency checking
    b. Plan recognition
4. Recognizing stereotyped errors: Requires knowledge of symptoms related to known bugs.

Those are just four “global” debugging strategies recognized in the literature. After conducting a thorough review of literature concerning computational debugging with regard to the technical nomenclature of strategies and tactics, numerous words are used to describe any one strategy or tactic. This comes down to a matter of semantic, linguistic, and authoring styles when describing or defining a strategy or tactic. To consolidate the variation within the literature, the researcher has grouped each strategy and tactics by categorizing each method by similarity. Refer to, *List of Debugging Strategies & Tactics* in Appendix A, for a synthesis of high-level debugging strategies and low-level tactics extracted from the literature on debugging strategies and tactics.

While various debugging strategies and tactics are often taught in programming classes, the application of debugging is learned by experiencing the errors and figuring out ways to correct them over time (Gugerty & Olson, 1986). Similar to the variation in terminology, used within the literature, to describe the strategies and tactics, there is also an unlimited amount of errors that one can potentially experience while programming. It is expected to encounter a considerable amount of errors when programming, therefore a considerable amount of concentrated problem-solving is required to eliminate errors (Gugerty & Olson, 1986).

Types of Errors in Debugging

A positive attitude towards errors and the debugging processes is helpful for keeping motivation and confidence for coding (S. Fitzgerald et al., 2010). Program bugs can be classified into two categories syntactic bugs, and non-syntactic bugs (conceptual, logical, semantic) (Gould & Drongowski, 1974). Some of the errors observed in the literature are syntactic (Badiozamany & Wang, 2010; Falahah et al., 2015; Kessler & Anderson, 1986; Klahr & Carver, 1988; McCauley et al., 2008; Romero et al., 2007), semantic (Badiozamany & Wang, 2010; Kessler & Anderson, 1986; Klahr & Carver, 1988; McCauley et al., 2008; Romero et al., 2007), systemic (Alqadi & Maletic, 2017; Gugerty & Olson, 1986), and correspondence errors (Nehaniv & Dautenhahn, 2002).

Syntactic errors are common and are errors in the sequence usually caused by a typo or miscommunication (Ahmadzadeh et al., 2005). They are categorized as a structural error within the code. Some of the syntactical errors can be confusing as an early coder from a natural language and a formal language prospective. As humans, if we come across a syntactical error within the written English language, the error does not completely hinder us from understanding the verbiage. If a sentence does not have a capitalized letter at the beginning or a period at the end, we can still conceptualize the concept. Robots, however, cannot compute code with a syntactical error such as an absent start or end block.

During the beginning stages of coding, it is common to produce and debug syntax errors as you are learning the coding language. Previous studies of children revealed that less skilled readers were significantly slower at low- level skills such as letter and word encoding (Wiedenbeck, 1985). The low-level skill of letter and word encoding in reading are similar constructs to syntax errors in debugging which provides further research to examine if a possible relationship exists. After frequently experiencing these errors, the majority of the superficial syntactic errors are easy to find which can lead to expert debuggers autonomously debugging them (Gugerty & Olson, 1986). There are structural errors that are embedded in the code and there are also errors within the context of the code.

Semantic errors are considered invalid program logic which leads to producing an incorrect result when the program is run (Ahmadzadeh et al., 2005). This type of error is associated within the context of the program. The most common semantic error seen in coding is the failure to define a variable (Ahmadzadeh et al., 2005). By not defining a variable the code can still operate, but it will not result in the intended response. The programs may be functional, but not the program the coder intended to write for a specific purpose. Transposed to an early childhood perspective, this would be equivalent to misusing a directional block. Children are often still learning their left from their right which can result in the misuse of one in place of the other. These errors can be considered trickier than syntax errors because the code itself is an operational code. However, it requires the coder to work backwards from the output of the program to attempt to identify where the logic was not met (Ahmadzadeh et al., 2005). Not all coding errors are created by the programmer while they're writing code. Errors can become apparent based on the coding platform itself.

Systemic errors occur within the interconnected parts of a system. These errors are often described as the hardware level errors, which is often the case with KIBO the tangible coding interface. With a majority of tangible child friendly electronic toys, the incorrect attachment of the motors to the KIBO body would have the reverse action happening. The KIBO robot is considered a system with several moving parts that work together. If one of KIBO's motors incorrectly attached, it would cause KIBO to spin around. This happens as the correctly attached motor will compute the executed code, while the incorrectly placed motor with these errors are not considered in the physical structure or logic of the program, but in the coding instrument itself. Some errors observed within early childhood coding can be conceptualized by recognizing children are just beginning to conceptualize the difference between natural and formal language (Nehaniv & Dautenhahn, 2002). As they are learning the introductory concepts of written and verbal language, they are discovering how they communicate. Humans can more easily recognize corresponding variables, while software programs often fail to do so (Nehaniv & Dautenhahn, 2002).

Correspondence errors refer to the problem of identifying which parts or items coincide with others (Nehaniv & Dautenhahn, 2002). The process of understanding correspondence is a cognitive developmental milestone as it builds on the foundational skills for later academic success (Bers, 2018). Infants begin to learn correspondence ideas while producing behaviors similar to their mother. Mimicking or imitating can be considered at some level a correspondence by the ability to observe a behavior and copy it (Nehaniv & Dautenhahn, 2002).

Transferring this concept to computer programming, some tangible coding platforms require sensors to complete functions within the code just as humans would need senses to complete similar actions. The KIBO body is comparable in respect to a human body as the programmer needs to add the attachments for the body to be complete. If humans did not have ears, we wouldn't be able to hear. The same concept is transferable to the KIBO robotics platform, as will become clear in the passages that follow. For example, if KIBO does not have the ear attachment, then it cannot compute a command such as the "Wait for Clap" block. The KIBO will not complete the code if the programmer does not correspond that for the robot if it does not have an ear it cannot hear to go to the next function. This reinforces that hardware and software are interconnected parts of a system (Bers, 2018).

Programmers develop approaches to analyzing and debugging a variety of problems as they are exposed to them through experiences and while they become more advanced in their programming skills (Jeffries, 1982). Exposure to problem solving can be seen in a variety of domains beyond debugging. Constant exposure to debugging provides the opportunity to grow the programmer's toolbox (Perkins et al., 1986), the same is apparent in the emotional development a child experiences, as they begin to understand and regulate their emotions. As children develop, they begin to grow their toolbox of emotional problem solving (Harris & Saarni, 1991).

Emotion in Early Childhood

The following section will touch upon a few key theoretical underpinnings of early childhood development as they relate to emotion. Topics considering emotion as a factor in early childhood development, decision making and problem-solving, modes of expression (ex. facial expressions), and intrapersonal as well as interpersonal understanding of emotion are discussed.

The adjustment and transition into formal schooling is a notable milestone for children with regards to emotion (Kostelnik, 2012; Denham, 1998). This is a period of time where novel social experiences, or opportunities to learn from one another, are introduced to children. Each opportunity inevitably leads to some form of emotional adjustment or observation as children interface with their peers (Denham, 1998; Graziano et al., 2007).

A new dimension of emotion is potentially imparted on children as they navigate and acclimate to a new academically based social landscape. In this setting children learn social and emotional strategies (Denham, 1998; Kostelnik, 2012). Peer-to-peer social interaction fosters a testing ground to practice and observe early childhood emotion (Denham et al., 2013; Kostelnik, 2012). From a scientific perspective, this is a wonderful opportunity to examine the complexities of emotion from an early childhood perspective.

Emotion is a component of communication, expression, and is inherently intertwined with the process of knowledge acquisition in early childhood (Kostelnik, 2012). Observing children as they integrate with, and begin to navigate new territory (school), various facets of development become apparent (Kostelnik, 2012). More specifically, the interaction between children, in this new environment, become visible as children begin to form the basis of interpersonal and interpersonal understanding (Graziano et al., 2007). This is particularly interesting in the way that children exhibit or reveal facets relevant to this study, such as emotional regulation, attention, and behavior (Kostelnik, 2012).

Understanding Emotion

As children are introduced to an academic or school environment (pre-kindergarten to second grade), children generally do not initially possess a wide repertoire of emotional strategies (Harris & Saarni, 1991). This is a time when children encounter new dimensions of problem-solving, both emotionally and socially (Mayeux & Cillessen, 2003). Each interaction is an opportunity whereby a child can assess a variety of strategies (Mayeux & Cillessen, 2003).

In an experiment conducted by, Harris & Saarni (1991), one group of children were asked to think about a happy memory for 30 seconds, whereas a separate group was asked to think about a sad memory for 30 seconds. The two groups of children, as well as an unprompted group of children, were administered an unrelated memory test (Harris & Sammi, 1991). Results of the experiment found that children who were asked to think about a happy memory, performed better on the memory task than both the group of children in the "sad memory" group as well as the control group (Harris & Saarni, 1991). This provides evidence that emotion is correlated to task performance and memory. What's more, the findings suggest that type of emotion (positive vs negative) also has an effect on task performance and memory.

Within the realm of research on decision making, there is detailed work on the emotional development of children (Denham, 1998). Children must understand their emotions and the emotions of others in order to make appropriate decisions during social interaction (Denham et al., 2013).

Emotional Impact: Help & Hinder

A book written by Vohs et al (2007), meticulously provides a compelling case with regard to the emotional and cognitive factor of cognition, mainly presenting as influential with a rapid reactionary or reflexive force capable of effecting decision making. While Vohs et al. (2007) recognizes the interplay of emotion and cognition as critical factors influencing decision making, also mentions the influential extent emotion has, as emotion can generally manifest as an impairment, or prove to be beneficial to the cognitive process of decision making.

Vohs et al. (2007) asserts emotion as an important mechanism of providing cognitive feedback which in turn indirectly, and at times implicitly, influences behavior. Emotion can influence learning and behavior in a helpful or harmful way (Vohs, et al., 2007). Overall, Vohs et al., (2007) describes emotion as fundamental to the psychological basis of behavior.

Naturally, one can introspectively draw on one's own memory or human experience to find examples where negative emotions, such as frustration or anger, have led to negative or suboptimal decision making. Whereas happiness, while a pleasant experience, may have led to suboptimal decision making or otherwise effected decision making.

Emotions and Motivation

There are 6 universal emotions, projected physically and expressed explicitly with distinct facial expressions (Ekman, 1989). Ekman's (1989) research found each emotion as "universally" observed, meaning those expressions are seen indiscriminately regardless of race, gender, culture, or age. However, a wide range of emotion is experienced subjectively and interpreted or expressed linguistically in many different ways. For example, happiness can be described in numerous ways, varying by degree, such as elation or thrill, excitement. That example, reinforces the idea that any single emotion can be described on a spectrum, constrained by language, subjectivity, and context of use. Similarly, multiple emotions can be blended and defined conceptually, introducing additional dimensions such that subjective experience and the application of emotion can be described. For example, take the concept of "Hedonism", this is a conceptual school of thought that is defined by the application of two emotions, pleasure, and avoidance of suffering. For context, consider the emotion-based adjectives below, anger, frustration and enthusiasm (Ekman, 1989):

- Frustration is often a sub-form of anger expressed towards an unwanted stimulus. Relating this to KIBO, this could be exemplified as a child who verbalizes loud sighs.
- Anger is a physical or verbal expression of disdain.
- Enthusiasm is defined as an expression of excitement or pleasure.

These emotions can inspire, or be inspired by, other psychological states such as boredom, distractions, and persistence. Therefore, for the purpose of this study, the following are operationally defined as,

- Boredom is defined as general uninterest in a task.
- Distraction is defined as not displaying sustained attention on a task. For example, this could be imagined when a child is clearly not following activity. This can also be related to boredom, enthusiasm, or other factors.
- Persistence is defined as a heightened sense of focus, or continuous effort directed towards a single task. With KIBO, this would present as a child who is engaged with, and continuously, works through problems while interfacing with KIBO.

Addressing emotion is an important skill, especially when children are beginning to learn what their emotions are and why they occur. A child does not want to display an emotion and then be told “no, not now”, as this signals to children that emotions are bad and should not be expressed. Children are impressionable and need validation, not commands. On one hand emotion is a powerful a source of motivation (Vohs, 2007).

In order to achieve a more holistic picture of how children are debugging, and the role emotions are playing on the process, this research aims to provide literature on the growing area of debugging in early childhood. The previously mentioned literature provides information on debugging strategies and tactics, emotional impact to decision making, however limited to a population beyond early childhood development. Likewise, literature on child development provides the basis for theoretical understanding of emotional development. Therefore, the literature has provided a foundation to explore emotion while debugging, however limited to beyond the years of childhood development. Therefore, the goal is to shed light on:

1. What kind of strategies and tactics do early childhood coders use when debugging?
2. What role does emotion play during the process of debugging in early childhood?

Chapter 3: Research Study Design

The research study design, as well as all recruitment materials, consent forms, and surveys, were designed by the researcher and approved by the Tufts University Institutional Review Board (Protocol #1909027).

Participants

Participants were recruited by reference of the DevTech e-list, as well as the Eliot-Pearson Children's School (EPCS), both of which are active users of KIBO robotics. The DevTech e-list is not exclusive to EPCS and is inclusive of DevTech affiliated summer camp attendees and other DevTech related research project affiliates. The inclusion and exclusion criteria are outlined below:

Participant Recruitment Inclusion Criteria	Participant Recruitment Exclusion Criteria
• Ages between 5 and 7 • Any gender and ethnicity • Ability to understand and speak English (does not have to be the first language) • Typically developing between grades Pre-Kindergarten and 2nd grade	• Any parent or caregiver that refused written consent to audio recording.

Participants were selected and tested during February of 2020. 12 total participants were recruited. Participants were all children, age range spanned from five years of age to seven years of age, 5 years old (n=5), 6 years old (n=4), 7 years old (n=3). Parents reported their child's gender on the demographic survey which resulted in 6 male participants and 6 female participants.

Procedure

Each experimental session lasted 1-hour in duration, at a maximum. The duration of each experimental session began when a participant and their caretaker completed both the consent and assent form and stopped upon completion four required debugging missions, or when the maximum time allocated was reached, see Appendix B for protocol.

KIBO Robotics: A Tangible Coding Platform for Young Children

The KIBO robotics kit is a developmentally appropriate screen-free, tangible robotics platform for young children ages 4 to 7 years old. The KIBO robot has attachable motors, wheels, and sensors. Figure 1 presents the KIBO robot's embedded barcode scanner at the front of its body as it actively scans the wooden programming blocks. While scanning the barcodes, KIBO produces noise and flashes a green light to help the programmer know the scan was successful.

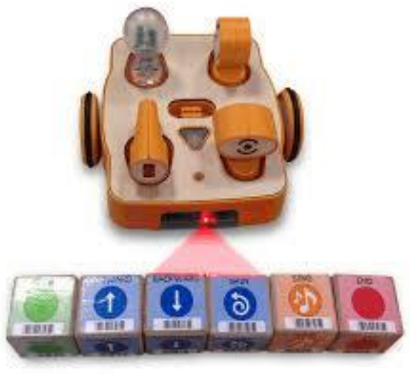


Figure 1. *The Tangible KIBO Robot Coding Platform*

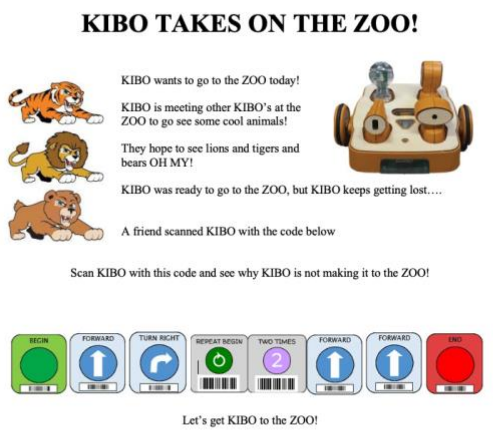
To program KIBO, a child must scan the wooden blocks with a green “BEGIN” block first and finishing the program with a red “END” block for KIBO to process the code. The wooden blocks are color-coded to designate their function; for example, orange blocks are sound blocks that produce auditory information, and blue blocks are motion blocks that can either move forward, backward, left, right, shake and or spin. Wooden blocks are designed to contain the word of the function, the color to indicate its purpose, and the images or icons represent the action. This provides a developmentally appropriate platform for children, who are still learning to read. KIBO has several added features that initiate play with peers and families (Sullivan et al., 2015).

The KIBO robot was chosen as the interface for this study for its interactive and tangible elements, which provide an essential platform for conversation during the debugging process involved in each Debugging Mission.

Debugging Missions with KIBO Robotics

The KIBO Debugging Missions were created as an activity to observe the debugging process in early childhood coding. Each mission contains one of the four different types of errors (syntactic, semantic, systemic, and correspondence) in order to observe the debugging process with a variety of commonly seen errors. They were read aloud to the participant and were designed with childhood themes of going to the zoo, the beach, home, train station, and school. Each mission contains a variety of coding blocks and a storyline to keep the participant engaged and enable a sense of helping the robot with a problem instead of the frustrating and difficult process while debugging code (Jeffries, 1982; Katz & Anderson, 1987; Kessler & Anderson, 1986; Perkins & Martin, 1986).

Based on the literature of commonly seen computer programming errors and commonly seen errors with KIBO, the missions were created to capture data on each major programming error type (ex. Mission 1: syntactic error, Mission 2: semantic, Mission 3: systemic, and Mission 4: correspondence). Example below of figure 2, for more, reference Appendix C: Debugging Missions.



KIBO TAKES ON THE ZOO!

KIBO wants to go to the ZOO today!

KIBO is meeting other KIBO's at the ZOO to go see some cool animals!

They hope to see lions and tigers and bears OH MY!

KIBO was ready to go to the ZOO, but KIBO keeps getting lost….

A friend scanned KIBO with the code below

Scan KIBO with this code and see why KIBO is not making it to the ZOO!

BEGIN
FORWARD
TURN RIGHT
REPEAT BEGIN
TWO TIMES
2
FORWARD
FORWARD
END

Let's get KIBO to the ZOO!

Figure 2. *Debugging Mission 1- KIBO Takes on the Zoo!*

The Debugging Missions of each experimental session provided qualitative data regarding insight into each of the participants debugging processes. Each mission was analyzed by metrics of time, completion, use of tactics, and use of strategies. Participants verbal and physical interactions with KIBO to solve the error were recorded. The actions and words used by participant during each Debugging Mission led to an in-depth analysis of the participants debugging process with different types of errors. The data were collected during the debugging missions and served to evaluate the research questions.

Debugging Mission Phases

The format of each experimental session involved three phases. The first phase, or "Introduction", required the child and their parent or caretaker sign a consent and assent form.

Parents or caretakers completed a demographic survey. The second phase was one in which each participant engaged in the experimental procedures, which included the four to five Debugging Missions along with semi-structured questions. The third and final phase, concluded the session. This occurred in a semi-structured format, to acquire information about the participants overall experience, challenges, thoughts, opinions, or stories.

A detailed account of each phase, phases one through three, went accordingly.

Phase 1: Introduction

During this phase of the study, a parent and child completed a consent and assent form, each were stamped with IRB approval. Each parent completed a 12-item pre-survey, which contained questions regarding their child's demographic information (gender, age, grade level, languages, etc.), and prior KIBO experience. Each parent completed the pre-survey. Two video cameras were set up in the room based upon the response to the consent form, they were used for either audio or video purposes. One camera was set up in the corner to observe a large area within the room and the other was set up closer to where a child engaged in the experimental session.

Phase 2: Study Procedures

The parent filled out the pre-survey, and the child was instructed to begin the study activities. The child was asked to participate in the KIBO: Debugging Missions (see Appendix C). Each of the missions contains a scenario where the child will need to debug one of the four previously mentioned types of errors syntactic, semantic, systemic, and correspondence. While the participant is completing each mission, the researcher will be prompting the participant with questions in order for the child to verbalize their thought process and emotions for later qualitative analysis.

Phase 3: Conclusion

At the end of the experimental session, the researcher informed each participant to complete a wrap up and reflection on the Debugging Missions. This phase was brief to identify missions that were perceived as more difficult, and to address any emotions that may have been felt during the process. Afterwards, the researcher transcribed the semi-structured qualitative data. Each experimental session involved recording of video and audio.

Observational Conditions

The researcher conducted semi-structured interviews to elicit unbiased information from each participant. Participants responses were recorded to understand how the process of debugging each mission went and served to expound upon any emotions the participant felt during the 1-hour experimental session. This conversation provides qualitative data as to how the child explain how they felt about the problem-solving process and if they displayed emotions to what degree that affect them. It is expected that qualitative data will allow the researcher to observe any overarching debugging strategy that may be derived from the tactic's participants used.

Chapter 4: Methodology

The following experimental methodology was used when analyzing both aspects of data collected, qualitative data and quantitative data, with metrics outlined below. All qualitative data was first transcribed, then reviewed, and processed manually. Data were aggregated, analyzed, and synthesized, pulling from each transcription to capture attributes of each metric listed below. One camera was used as the main reference point for each participant. That resulted in all time stamps taken from one camera and the other camera was used for added clarity (facial expressions, physical movements, etc.).

Qualitative Metrics

Qualitative data was extracted from the transcript of each Debugging Mission. The framework used to address each research question involves targeting (1) debugging approaches and (2) displays of emotion.

1. *Debugging Approaches (Strategies and Tactics)*

In the transcribed data, each participant attempt to debug was coded as an "approach". To eliminate bias, all approaches were collected from the data and were later classified into the categories of strategies and tactics. This provided a holistic picture of the debugging process to be synthesized into cohesive summary.

2. *Displays of Emotion (physical – facial expression, body movement; verbal -- noises, statements)*

Displays of emotion contained all data with physical responses of facial expressions and body movements as well as the verbal responses from the participants such as noises and statements. In order to get a holistic picture of the emotions, both the physical and verbal displays of emotion were noted to strengthen the identification of emotions. Usually multiple expressions of emotion happened at the same time (ex. Hands thrown up, saying something with a tone that sounds frustrated, and facial expression looks confused, upset, frustrated). This provided multiple timestamps and a clear identification of the emotional feelings. In order to assess and code the transcripts for qualitative data, each category noted above provided data to answer the research questions.

Quantitative Metrics

The quantitative data collected in order to answer the research questions are listed below as four variables:

1. *Listened to Debugging Scenario*

This metric is operationally defined as either:

  a. A child did listen to the Debugging Mission Scenario
  b. A child did not listen to the Debugging Mission Scenario

Listened to debugging scenario involved the participant engaging with the Researcher as they read the mission paper for each debugging missions. As the scenarios are playful stories of KIBO on an adventure, the last sentence of each mission provides a phrase to structure the participant by having them "scan the code to see why it's not working". The prompt provided structure for the participants to engage in the mission. By listening to the mission, there was a chance that the background knowledge of the story would provide context and support for the participant to debug the mission. Some children were still getting acclimated to the situation and to instruct the participant to solve the error could have been too unstructured for such a young age group, children ages 5 to 7-years old.

2. *Mission Start and Stop Time*

Start time was operationally defined as any instance when a participant verbally or physically signals intent to solve mission (ex. picks KIBO up, changes code blocks, says what might be wrong or what might be needed). Stop time was operationally defined by criteria listed below:

  a. The mission is solved with the participant solving the error resulting in KIBO at the end destination (KIBO completes code at end destination).
  b. Participant stops working on the mission (ex. Runs out of time, gives-up, wants to move on, etc.)

3. *Length of Time on Mission*

Length of time spent on mission is a metric based on start and stop time which included if participants solved the mission or the amount of time participants spent until giving-up or otherwise, signaling a stop time.

4. Mission Completion

"Mission completion" is operationally defined as KIBO programmed from the "Start Marker" reaching the end destination for each Debugging Mission. Mission completion data were collected in order to measure the participants successfulness of solving the mission error or lack thereof.

Chapter 5: Results

Qualitative Results

Qualitative data culminated into categories such as emotion, mental models, and approaches to debugging. In doing so, data were extracted from transcriptions of each experimental session, and analyzed thematically by grouping similar results, per participant, across Debugging Missions. For a comprehensive view, see Appendix D. Therefore, the intention of this section is to provide a high-level synthesis of data produced by all experimental sessions by category recorded.

Emotion

Emotion was an attribute clearly observed, and noted, during each experimental session, across all participants and Debugging Missions. Emotion presented in two forms, physical expression (ex. facial expression, body movement, gestures) and auditory expression (ex. verbal statements, noises, utterances). Furthermore, participants displayed a range of emotional polarity from positive (ex. overt happiness and excitement) to negative (ex. extreme frustration and anger).

Participant 3 and Participant 11, both age 5 years old, displayed emotion that is representative of the psychological state of "extreme frustration". Each participant's performance was poor, as both Participant 3 and 11 each completed only the first Debugging Mission. To illustrate this point further, consider the case of Participant 11 starting with Debugging Mission 2. This participant first displayed "frustration" only moments into Debugging Mission 2. The participant stated, "done", in an unhappy tone while turning the wheels of KIBO in a frustrated and aggressive manner. Extreme frustration was later revealed during Debugging Mission 3, when Participant 11 stated the desire to move on, "how about we can try that mission (while pointing to another debugging mission paper) … that mission up there (continues to point to location of debugging mission paper)". Participant 11, then went on to begin Debugging Mission 4, which concluded by the participant rolling onto to the floor, with their back against the ground, and arms extended outward, stating "let's just skip this one".

Participant 3 also exhibited extreme frustration, which first occurred during the onset of Debugging Mission 1. This gradually escalated to a physical display of anger during Debugging Mission 3, wherein the participant physically hit KIBO. Subsequently, this behavior was recorded as "anger" and "extreme frustration". For reference, see figure 4, pictured below.

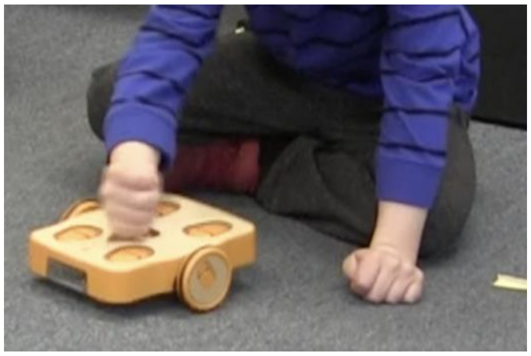


Figure 3. *Participant 3 Debugging Mission 3*

Participant 10 exhibited a range of both negative and positive emotion, throughout the duration of the experimental session and within each Debugging Mission. While a range of emotion was displayed, this did not prevent Participant 10 from completing any of the Debugging Missions.

During Debugging Mission 4, Participant 10 begins “impulsively” by pulling the mission paper out of the way, without listening to the researcher, and begins to immediately start scanning the code blocks. The participant then physically forces (ex. “aggression” or “excitement”) KIBO to move by pushing KIBO with their hands, then the participant claps in “joy” as KIBO is drawn closer to the end destination. It is important to note this participant is five years of age.

On a positive note, the majority of participants displayed “happiness”, in one form or another (ex. “excitement”, “encouragement”, “joy”), when KIBO successfully reached its end destination, which concludes the Debugging Mission. As an example, see figure 4, pictured below.

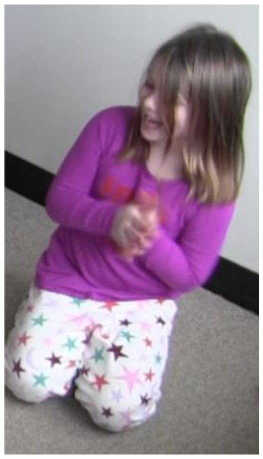


Figure 4. *Participant 7 Debugging Mission 1*

Another display of emotion observed, include the moments of encouragement that the participants vocalized towards the KIBO robot. Participants often made statements of affirmation towards the KIBO to motivate and reassure the KIBO of its adventure to get to the end destination. Participant 0 and 7, engaged in the encouragement of the KIBO throughout all of the missions, with statements such as “Come on KIBO, you can do this!” and “Wow, Good Job KIBO!”.

Participants experienced a level of distraction, either within the mission by creating stories or jokes about KIBO, which led away from the debugging, or with other comments, stories, and such (ex. sticky notes, talking about ski trip, discussion of friends in class, and stories about their life). These distractions resulted in disruptions of the Debugging Mission as participants focus on the task at hand was derailed by their thoughts. For example, Participant 7 corrected the error in a mission, but before they could scan the new code, they told a story about a ski trip they were going on soon, which extended their length of time on mission and delayed mission completion. Brief separation of focus on the Debugging Mission, representative of “absentmindness”, which delayed participants from scanning new code and also caused particiapnts to run KIBO with an older or different code. This inevitabley led Participant 7 to display a “confused” state until she was able to assess and recognize the code output did not match the code blocks.

Distraction varried by degree and appeared amongst several participants which displayed a variety of result, mainly in the form of “frustration” and “boredom”. While, others were distracted as a result of the overwhelming emotions felt the participant. For example, Participant 10 expressed heightened emotions when transitioning from Debugging Mission 3 to Debugging Mission 4, which manifested as distration and clearly was a factor in limiting their ability to focus on Debugging Mission 4.

Additionally, indicators of “excitement” were observed as participants displayed joy, giggles, physically jumping around, and facial expressions directly related to “happiness” and “excitement”. For example, Participant 10 could not contain their emotions and became distracted by telling stories about KIBO, and created elaborate scenarios where KIBO would knock down the other end destinations. These behaviors were not observed earlier, as in towards the onset of each mission. And were objectively uncharacteristic of the participant, which made these observation notably different. For example, Participant 10 was easily distracted due to the exicted state they were experiencing. However, distractions were a common theme displayed by Participant 11, as every mission they demonstrated distration, due to objects in the room, activity outside of the window, or movement by people outside the setting of the experimental condition. Furthermore, the distractions Participant 11 experienced, increased as they became “frustrated” and “bored” with the Debugging Missions. For example, Participant 11 was distracted consistently throughout each Debugging Mission by a pack of sticky notes which they insisted on returning, or commented on during each mission.

Mental Models

Each participant revealed insight into common mental models as they actively debugged each mission. For example, all 12 participants reference the KIBO robot with the pronoun “he”, an interesting word choice that demonstrated an implicit gendering of KIBO. Furthermore, participants engaged in treating KIBO as a human by giving KIBO human-like characteristics. For example, Participant 7 expressed concerns for the KIBO when KIBO did not do what was expected. A participant used the phrase “he doesn’t feel well” to explain KIBO’s confusion or inability to execute code. This is a prime example, where Participant 7 both assigned KIBO a gender with the pronoun “he”, and associated KIBO with human traits such as sensation and human vulnerability to illness (ex. sensation “feel”, vulnerability to illness “doesn’t feel well”).

Additionally, Participant 4 thought KIBO did not understanding the participant so the participant positioned their body close to KIBO and proceeded to shout, “Hey KIBO, did you get that?”, to both ensure KIBO would remember the code and to pose a question to KIBO in a way typically representative of a human to human linguistic exchange.

Approaches to Debugging

Through thematic analysis the following approaches to debugging became apparent amongst participants. This section is an exhaustive list of each approach with examples and cases demonstrated by participants.

Methodical Reasoning is the act of thinking closely about decisions, which is the opposite of acting quickly without reason. Several participants (n=4) engaged in the strategy of methodical thinking throughout various missions. Methodical thinking was displayed in a variety of ways, but primarily as a process of reasoning stepwise through the code to debug an error. Participants typically provided an explanation for their reasoning, acted on the idea stated, checked by responding to feedback (KIBO output or researcher output), and then adjusted accordingly. For example, Participants 0, 1, 2, and 7 displayed this strategy throughout their Debugging Missions.

Scanning was noted as an approach, and best exemplified by Participant 4, who performed this strategy throughout all missions. Participant 4 consistently separated the block code and scanned one block at a time when programming KIBO. For example, Participant 4 would take a block from a pile, scan it, and then move it to another pile of blocks so to separate scanned blocks from unscanned blocks.

Re-scanning, this approach was observed throughout all missions and across all participants. This approach was most notably observed during Mission 3 as most participants checked the code and adjusted accordingly in response to KIBO’s output.

Adjust to feedback, participants responded to KIBO either from the error message or output KIBO provided. After running the program or guessing the origin of the error, all participants made adjustments based on the feedback provided by KIBO (ex. the beeping noise KIBO makes when an error occurs during scanning).

Chunking or slicing, involves reducing the problem space by using smaller pieces of code**,** to complete the mission or chunking out portions of the code. Two participants used this approach, which entailed utilizing smaller pieces of code to program KIBO to the end destination. KIBO ran the program by coding and incrementally adjusting to KIBO output to run the KIBO’s next code to reach the Debugging Missions specified end destination.

Block Connection, the majority of participants encountered this circumstance. This is a circumstance where blocks did not fit together, due to the way blocks are connected, which in turn prompted the majority of participants to try something different or adjust their approach.

Guessing, the majority of participants provide an explanation to the actions they took while guessing. However, others made guesses and either immediately acted on them or only stated the guess but provided no explanation or reason for the guess.

Hardware, each participant checks KIBO’s hardware by physically inspecting KIBO at least once during each experimental session. Some participants added sensors to KIBO for no reason, while others added sensors incorrectly but provided rationale for the use of sensors. Sensors were only necessary in one mission, Debugging Mission 4, which needs an ear sensor. Overall, the approach of adding sensors was exhibited by the majority of participants throughout each mission. At least once, any given participant vocalized, in the form of a guess, that the error could be due to a hardware problem such as missing a sensor. This is best seen by participant 4 who constantly started each mission by adding all of the sensors in an indiscriminate fashion.

Tinkering, for example, one participant solved Debugging Mission 1, which involved using a repeat block. The participant was able to circumvent the problem without knowing the function of the block. This was achieved by serendipitously placing the block into the correct sequence of code, and surprisingly discovered that the repeat block had worked.

Walking out the code in an attempt to locate the bug, some participants physically moved either their bodies by walking or crawling, or moved KIBO, in a way to visualize the anticipated behavior of the code.

Forcefulness, Participant 3, 8, and 11, all demonstrate this strategy. This is best depicted by participant 3, who stated “I’m just going to push him to the zoo because it’ll never get there” upon feeling frustrated. This strategy is an attempt to physically override or circumvent the program altogether by physically picking KIBO up or pushing KIBO to the end destination.

Asking clarifying questions, all participants asked questions relating to KIBO’s actions, parts, or code blocks so to check or reaffirm their own conceptual understanding of any respective component of the Debugging Mission. Of course, participants asked unrelated questions, however, there were questions that were approach-related with regards to the mission. For example, participants asked about items they were uncertain of, such as KIBO’s internal wires, KIBO’s hardware, the Debugging Mission in general, or if there was a certain way something had to be done. For example, Participant 10 asked if they still pushed the triangle the same way to start KIBO, Participant 10 then asked if this was the intended code per the mission.

Tracing the code, this approach was demonstrated by a few participants (n=4) who physically tapped each block of code, so to follow the function of code or internally reason through their own thought processes, determining if the code would suffice to bring KIBO to the end destination**.** Participants demonstrated this in a variety of ways (ex. physically tapping their fingers on each block, walking the code out or vocally describing the expected path. Two participants even drew the path while speaking, regardless of ability to read or write).

Deleting code to start over with new code were an approach taken by Participant 5 and 9 as they decided to not use the original mission code. Participant 9 did not use the original code during one mission because they didn’t believe the code on the mission paper and the block code were the same but was only able to speculate some extra lines on the block as the issue. Participant 5 provided the explanation that they didn’t think it would work for completely changing the original code in one of the missions.

Seek feedback from Researcher was an approach across all participants as they looked to the Researcher for feedback on KIBO’s output and the participants guesses for the location of the error. This approach was similar to what would be experienced in a classroom with a teacher.

Removing unknowns, this approach appeared from Participant 0, 1, 2, and 4 as they expressed an unclear understanding of the blocks meaning. The participants removed the block to eliminate the unknown factors in the code.

Running the program entailed pushing KIBO’s △ which would signal KIBO to run its program. All participants ran programs as they worked on the mission, but Participant 3, 6, 8, 9, 10, and 11 were persistent in running the program multiple times. Running the program was often seen as an attempt, which involved pushing the green triangle △ on KIBO, which triggers KIBO to execute its program and begin moving. Pushing △ multiple times even when the light was not flashing green signaled that KIBO had a program in memory, loaded, and ready to run. The act of “Running the program” required participants to continuously return to the green triangle and push the unlit button, in an attempt to make KIBO go (execute any program loaded into KIBO).

Connections and associations were apparent in the case of three participants, who were each 7 years old. Each participant made associations to prior knowledge, at times referencing a previous Debugging Mission or otherwise connecting mission circumstances to entirely different concepts (ex. associating a musical scale with the number of blocks). Participant 2 and 7 connected the current Debugging Mission they were working on with a Debugging Mission that was previously completed. During Mission 3, Participant 2 and 7 referred to Mission 1 as they connected ideas of how KIBO's output in Mission 1 was close to the end destination for Mission 3. Interestingly, Participant 0 connected a musical scale to the number of KIBO code blocks in Mission 1. For example, Participant 0 saw the 8 code blocks and then stated, "I just realized something" and sang "do-re-mi-fa-so-la-ti-do" while tapping on each individual block.

In addition to the aforementioned approaches, themes were also derived from the data. Themes across participants are labeled as "mis-scanning", "identify the problem yet unable to apply a solution", "uncertainty", "concept vs. application", and "batteries".

Mis-scanning was a common theme amongst most participants which resulted in the participant adding extraneous errors or problems to the code, which was overall unproductive in debugging each Debugging Mission error. Interestingly, this technique was successful for one participant. Participant 10, mis-scanned blocks during Debugging Mission 1. Which surprisingly led the participant to correctly delivering KIBO to the intended end-destination. This was a fortuitous accident, which supports the idea that sometimes computer programmers debug problems without understanding the root cause or correlation of their change and the success of the overall outcome.

Identify the problem yet unable to implement solution. All participants noted, when prompted by the researcher, if the error was a problem with KIBO (ex. hardware, sensors, etc.) or a problem with the code (ex. missing repeat blocks, end blocks, etc.). Most often the participants accurately identified the general problem area with the code, conceptually (ex. Debugging Mission lacks a repeat block, participants would state, "KIBO needs to go forwards four times"), but were not able to follow through with executing it (ex. participants did not make the connection to using a repeat block to move KIBO forward four times). Participants would continue to encounter the same situation and pick the correct area in the code again but would continue to struggle to follow through with implementing a solution. Participant 1 expressed verbally how to fix the error in Mission 1 but could not apply that knowledge to the code as they continued to circle back to try other tactics.

Uncertainty was a theme characterized by the use of words such as **"Maybe"** and **"I don't know"** were terms used often when a participant was out of approaches or ideas. The addition of the word "maybe" was commonly seen when participants, with an uncertain tone, would continue to guess incorrect solutions as they attempted to debug. Participants responded with the phrase "I don't know", generally at times as they ran out of approaches to solve the problem. The use of "I don't know" is best displayed by Participant 11, as they resorted to using the phrase consistently and quickly without providing thought or meaningful response throughout each Debugging Missions. It is important to note Participant 11 generally had a frustrated demeanor and continuously gave up on each mission.

Concept vs. Application, some participants understood the function of each block but were unable to apply the blocks correctly to solve the mission. Whereas, others were able to apply or use a block correctly in the logic but confirmed no knowledge of the blocks function. In some cases, the participant knew neither what the block did or how to apply it.

Batteries was a guess provided by all participants, almost reflexively as an explanation as to why the KIBO would not work as intended (ex. the program would not run). The majority of the participants stated "batteries" as a guess at least once during a session, however a few participants continued to default to guess "batteries" multiple times whenever KIBO displayed an error. Participant 11 continuously guessed that batteries were the source of the error, and rarely deployed any other tactic or effort to debug the error further.

Clearly, there are a plethora of approaches that were observed amongst participants. The table below provides a list of each approach in no particular order.

Table 1. *Participants Approaches to Debugging*

Methodical Reasoning	Guess	Tracing the code
Scanning	Hardware	Deleting code
Re-scanning	Tinkering	Seek feedback from Researcher
Adjust to feedback	Walking out the code	Removing unknowns
Chunking or slicing	Forcefulness	Running the program
Block Connection	Asking clarifying questions	Connections and associations

In sum, the most notable approaches included, "Methodical Reasoning", "Guessing", "Hardware", "Walk out the code", "Connections and associations", and "Block Connection". This also led to formulating themes across participants such as "Mis-scanning", "Uncertainty", "Concept vs. Application", and "Batteries".

The most common approaches to debugging are depicted by “Methodical Reasoning” which was used by Participant 0, 1, 2, and 7, throughout all of their missions. These participants used the approach “Guess” to first form assumptions as to why KIBO would not work. Then each participant would provide rationale in support of each guess, take action according to the guess, and then adjust per feedback to continue iteratively until solving the mission. These participants were in the higher range of the age group (P0, P2, P7 are 7 years old and P1 is 6 years old). Additionally, Participant 4, also showed a consistent approach across missions. Participant 4’s approach involved “Hardware” and “Scanning” indiscriminately.

The aforementioned examples serve to illustrate the primary cases exemplar of debugging approaches, and themes, that were revealed throughout each experimental session. In sum, a variety of approaches and themes were documented; a total of 18 approaches were observed and five overarching themes were distilled from the qualitative data collected. The variation in approaches children engaged in while debugging is similar wide variety of strategies and tactics described in the debugging literature, see Appendix A. observed in populations beyond early childhood.

Quantitative Results

Quantitative data was measured by metrics of length of time on mission, the start and stop time during mission, mission completion, and listened to mission. Participants length of time on mission is displayed below, see figure 5.

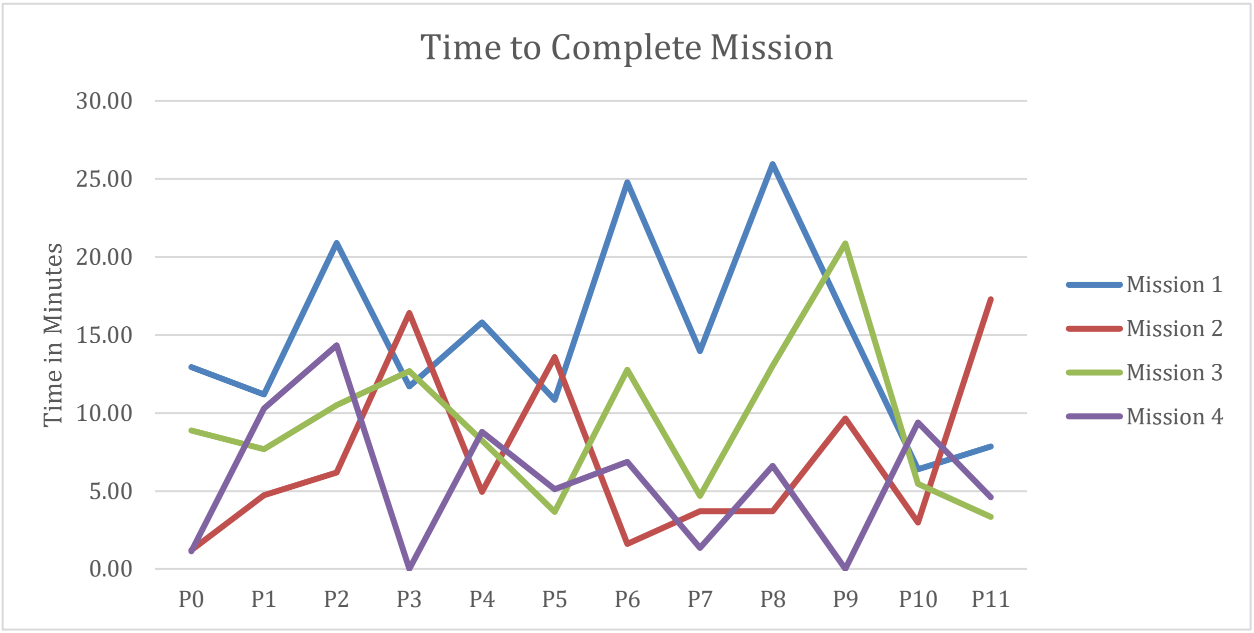


Figure 5. *Length of Time on Mission*

Participant 6 and 8 spent the majority of their time on Debugging Mission 1. Participants 3 and 5 spent close to the same amount of time on Debugging Mission 2 as they did on Debugging Mission 1. Participants 5 and 7 spent the shortest amount of time on Debugging Mission 3, where Participant 9 took the longest time (P9 time on mission > 20 minutes). Participants 2, 4, 8, and 10 spent more time on Debugging Mission 4 as opposed to the other participants. It is important to note the fluctuation of time spent on mission, per participant, and across Debugging Missions.

Table 2 displays the *Listened to Debugging Mission* data for each participant across the four Debugging Missions. Each participant was given the opportunity to listen to the Debugging Mission for each mission and as noted in the table, a majority of participants listened. Participant 0, 1, 2, 4, 7, and 9 listened to the instruction read aloud by the researcher as it was written on each Debugging Mission paper. Participant 11 was not engaged and did not listen to the Debugging Mission, up until the final mission. During Mission 4, Participant 3 and 9 did not receive scores as they did not participate in Mission 4. Participant 3 gave up and did not continue on to Debugging Mission 4. However, Participant 9 ran out of time during the experimental session which resulted in the participant not being able to start Debugging Mission 4. For reference, see table 2 below.

Table 2. *Listened to Debugging Mission*

	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
Mission 1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Mission 2	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	Yes	Yes	No
Mission 3	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	No
Mission 4	Yes	Yes	Yes	N/A	Yes	No	Yes	Yes	No	N/A	No	Yes

Table 3 displays the *Mission Completion* data for each participant across the four Debugging Missions. Participant 0, 1, 4, 5, 6, 7, 9, and 10 completed the missions as they were operationally defined. However, Participant 11, completed Mission 1 and gave up on Missions 2-4. Participant 3 completed Mission 1 and also gave up on the remaining Missions, but Participant 3 asked to stop after Mission 3 as did not want to continue. Participant 8 did not solve Mission 1 or 3, as the participant became increasingly aggravated, frustrated, and physically pushed the KIBO to the end destination without programming. Participant 9 completed Missions 1 through 3 but ran out of time and was unable to start Mission 4. For reference, see table 3 below.

Table 3. *Debugging Mission Completion*

	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
Mission 1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Mission 2	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Mission 3	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	Yes	Yes	No
Mission 4	Yes	Yes	No	N/A	Yes	Yes	Yes	Yes	Yes	N/A	Yes	No

Chapter 6: Discussion

Debugging is central to coding because in the course of programming, errors are inevitable. Four major types of errors occur, namely, syntax and semantics as well as systemic and correspondence errors. In the literature on debugging, what is often overlooked are the emotional underpinnings of programmers when confronted with the task of debugging. There is a general lack of literature that focuses on the approaches to debugging in early childhood, therefore the goal of this experiment set out to answer the following research questions.

Research Question #1: What kind of strategies and tactics do early childhood coders use for debugging and solving errors?

Therefore, data were analyzed to gain perspective in terms of approaches early childhood coders use when debugging using the KIBO robotics coding platform. The most notable approaches exhibited included, “Methodical Reasoning”, “Guessing”, “Hardware”, “Walk out the Code”, “Connections and Associations”, and “Block Connection”. A comprehensive list of approaches with examples per participant is provided in Chapter 5: Results. Additionally, themes emerged across participants such as “Mis-scanning”, “Uncertainty”, “Concept vs. Application”, and “Batteries”. Those themes provide additional perspective and insight into the way’s children debugged and problem solved errors within the KIBO robotics platform.

To that end, participants took the approach of “Methodical Reasoning” when debugging (Participants 0, 1, 2, and 7), throughout each of their missions. Within this approach, those participants also used the approach, “Guessing” to first form assumptions as to why KIBO would not work. Then each participant would provide rationale in support of each guess, take action according to the guess, and then adjust per feedback to continue iteratively until the mission has been solved. The “Methodical Reasoning” approach demonstrated by the participants directly relates to the debugging strategy and cognitive process of “Thinking” (Deliema et al., 2019; Fitzgerald et al., 2010). “Thinking”, in the literature, is described as the mental process of focusing on the problem while speculating the possible cause of a bug (Deliema et al., 2019; Fitzgerald et al., 2010).

Furthermore, within “Methodical Reasoning”, participants engaged in the act of “Guessing” as to why KIBO would not work, then providing rationale in support of each guess. However, other participants also engaged in “Guessing” but did not provide rationale or reason to support the guess. This form of guessing is directly related to the literature on “Guessing”, which is defined as providing no reasoning, or attempting without evidence (Fitzgerald et al., 2005; O’Dell, 2017).

Participants also demonstrated the approach of “Hardware” at least once during each experimental session. Almost all participants guessed, at least once per experimental session that the root cause of error may be due to KIBO’s hardware, including but not limited to the sensors. Only one Debugging Mission required the addition of sensors to solve the error. The type of behavior participants exhibited when taking the approach of “Hardware”, is closely related to literature on debugging approaches, specifically “Just in Case” (Fitzgerald et al., 2010; Murphy et al., 2008). “Just in Case” involves unnecessary changes, such as fixing brackets, or additional parenthesis, which may or may not always be the solution (Fitzgerald et al., 2010; Murphy et al., 2008). This is similar to participants who engaged in the approach of “Hardware” as some participants, despite knowing that KIBO didn’t always need sensors to run, continued to advocate to add sensors at the start of the process.

These participants were in the higher range of the age group of the sample of 12 participants (P0, P2, P7 are 7 years old and P1 is 6 years old). Additionally, Participant 4, also showed a consistent approach across missions. Participant 4's approach involved "Adding sensors" and "Scanning" indiscriminately.

"Walk out the Code" was another noteworthy approach demonstrated by participants. A good segment of participants walked out the code either physically by walking or crawling, but also by drawing on the whiteboard, or picking up KIBO and placing KIBO along the intended path. "Walkthrough" is a debugging approach cited in the literature as talking through or emulating the code for understanding (Fitzgerald et al., 2005; Jefferies, 1982). This is yet another parallel between established literature on debugging in non-early childhood coders and the behavior exhibited by the participants.

"Connections and Associations" was another noteworthy approach demonstrated by participants, however, it varied in way it presented in participants. For example, Participant 0 made an unexpected associated between a musical scale and the number of KIBO code blocks in Mission 1. Participant 0 saw the eight code blocks and then stated, "I just realized something..." and sang "do-re-mi-fa-so-la-ti-do" while tapping on each individual block, which was fascinating. "Making Connections", as cited by the literature, is defined as relating previous experience or knowledge to a real-world experience (Deliema et al., 2019; Grigoreanu et al., 2009; McCartney et al., 2007).

"Block Connection" was another noteworthy approach demonstrated by participants. This was a circumstance where blocks did not fit together, due to the way KIBO blocks are connected, which in turn prompted the majority of participants to try something different or adjust their approach. While this does not align with any-one debugging approach noted in literature reviewed, it is due to a unique component of the KIBO robotic platform.

As we can see, the variation in approaches children engaged in while debugging is similar in wide variety of strategies and tactics described in the debugging literature regarding populations not inclusive of early childhood coders. This speaks to literature citing the fact that there is no “one size fits all” to learning computer programming and debugging (McCartney et al., 2007). Additionally, the experimental sessions provided a great basis for Papert’s theory of “learning by doing” because participants learned to debug errors in KIBO without being explicitly taught. For example, the participants in this study did not have access to resources that a computer programmer would typically have, namely, Stack Overflow. Nor did the participants in this study have the opportunity to ask for outside help, the help of a teacher, or receive any form of teaching from the researcher. This left participants entirely to their own devices, as the participants “learn by doing”.

The second goal of the research conducted was to answer the second research question.

Research Question #2: What role does emotion play during the process of debugging in early childhood?

As previously stated, understanding emotional reactions are especially important for understanding young children’s programming since managing emotion in the face of challenge is something still being developed throughout early childhood (Denham, 1998; Mayeux & Cillessen, 2003).

In the sample of five-to-seven-year olds, as expected, the older children often indicated they were managing emotions as they often asked questions, thought before acting, and were able to sit for longer periods of time on a task. Those participants were also more likely to exhibit the debugging approach of “Methodical Reasoning”. Whereas, younger participants were much less likely to listen to the mission paper before starting a Debugging Mission, and also demonstrated an increased level of impulsivity by posing guesses more frequently as well as handling KIBO in an excited or aggressive manner (ex. P3 and P11, both 5 years old). Asking questions and stopping to think are signals of managing emotion which is similar to emotional regulation. Interestingly, these are also considered a component of "Methodical Reasoning", which further illiterates that emotion and the debugging process are connected.

Younger participants also demonstrated frustration more often which resulted in giving up on the mission and displaying more angry outbursts. This is a clear indicator that the younger participants were less regulated with their emotions. The younger children were more likely to have outbursts which resulted in one child even hitting the KIBO, or they provided non-solutions such as adding a light "to see by" to the KIBO construction when a light was irrelevant to debugging.

This is also observed when considering the metric of length of time on mission as participants varied in the time, they spent on each Debugging Mission. For example, four participants (Participant 2, 6, 8, and 9) spent a longer period of time (>20 minutes) than the other participants on a Debugging Mission, but did not display overwhelming signs of frustration or anger but instead projected a sense of persistence or desire to keep going (ex. P9 says "I got this guys!" and provided affirmation by stating, "that was easy" once the Debugging Mission was completed). This demonstrates a relationship between the Length of time on Mission and the psychological state of "motivated (ex. "persistence") as those participants generally continued to persevere to solve the error and complete the Debugging Mission.

An additional finding became apparent as a result of the qualitative analysis. Participants revealed mental models such as encouraging or motivating the KIBO robot, assigning gender to

KIBO, and giving KIBO human characteristics. Participants were projecting a mental model of the male gender and human characteristic of “feeling well” onto the robot, KIBO.

Limitations

There are several limitations to this work. One of the first limitations is the sample size (n=12). However, due to the qualitative nature of this study required an in-depth analysis of each participant. These participants, as long as they meet the inclusion criteria and did not meet the exclusion criteria then they were not admitted into the study. The study was publicized through the DevTech e-list, and throughout the Eliot-Pearson Child Study and Human Development Department. Participants were recruited from the DevTech e-list (summer camps, past research projects, schools actively using KIBO) and Eliot-Pearson Child School (EPCS) which are projected to attract middle to high socio-economic status and highly educated families. The DevTech resources and enrollment in the Eliot-Pearson Children’s School are both considered privileged opportunities which may affect the generalizability of this study on a larger population of children ages 5 to 7 years old.

Another potential limitation pertains to the methodology and experimental design used. By analyzing both quantitative and qualitative data there was the possibility of several issues arising. Since Qualitative research is time consuming and labor intensive, especially accounting for coding the videos, the use of ethnographic or qualitative analysis software or forms of programmatic automation are recommended. Accurate interpretation of the video recordings and the interviews required extensive timing. Most of the questions and observations were open-ended and in a free forum which did not provide as much control to verify the responses as with quantitative research. There were also limitations for the quantitative research. The pre-experiment demographic survey relies on the self-reported data, which is not always reliable.

While this work may have limitations, it has the possibility to provide research for growing areas within the literature. Most of the research on debugging and the strategies used were conducted several decades ago and not with children or early childhood. There is little research on early childhood coders and their debugging process especially regarding the strategies they may use and if their emotions play a role in their debugging ability or approach. Providing this information can expand the literature and allow for future work on problem-solving strategies within debugging and identifying the role of emotions on debugging.

Future Directions

The results and methodology can be used as a foundation to further explore the area of emotion and computational thinking, specifically from an early childhood perspective. A potential outcome of delving deeper into this area may give way to driving momentum necessary to adjust the educational direction of teaching computer science, or computer programming, in early childhood.

The results and findings might potentially generalize to other applications where debugging is a necessary skill. Therefore, further research would provide value to understanding the basis as to how people approach debugging in the way that they do. Additionally, a benefit of achieving that level of understanding may shed light on an optimal way to address the variations exhibited in debugging approach. This may lead to improvements in curriculum development, or self-directed learning such as “Learning by doing” and so forth (Papert, 1993).

Further information is necessary to address debugging, from a wholistic perspective, especially in children that are emotionally vulnerable. What’s more, there is value in framing debugging such that it encourages the attitude of a growth mindset. Popular belief perpetuates the concept of debugging as difficult and time consuming, this can easily lead to promoting a negative connotation representative of a fixed mindset.

Emotions impact everything and it is part of human nature (emotions help us survive by alerting us of our body/mind reaction to environment/stimulus). How do we address emotions as we learn though? Emotions are not to hurt us but to warn us of potential dangers. Frustration shouldn’t be considered a bad emotion instead noted as a way our mind/body recognizes a potentially challenging situation of a breaking point.

Appendix A: List of Debugging Strategies & Tactics

Strategies & Tactics	*Definition*	*Supporting Sources*
Tracing (tracking)	The most common debugging technique of comparing the output with the code which can be done mentally, by hand, or with debugging tool. Analogous to forward reasoning program strategy.	(Alqadi & Maletic, 2017; Araki et al., 1991; Badiozamany & Wang, 2010; Chen et al., 2017; S. Fitzgerald et al., 2010; Sue Fitzgerald et al., 2008, 2005; Gould, 1975; Gould & Drongowski, 1974; McCartney et al., 2007; Murphy et al., 2008; Perkins et al., 1986; Romero et al., 2007; Vessey, 1985)
Hand Simulation	Looking for inconsistencies between what occurs in the actual program and what is expected to occur from the program	(Gugerty & Olson, 1986; Jeffries, 1982; Katz & Anderson, 1987; Klahr & Carver, 1988; Romero et al., 2007)
Walkthrough	Talking through some level of code emulation	(Sue Fitzgerald et al., 2005; Jeffries, 1982)

Tinkering	Making modifications that potentially could be random and unproductive. It can also appear from questioning or attempting to form a hypothesis of bug location	(Alqadi & Maletic, 2017; S. Fitzgerald et al., 2010; Sue Fitzgerald et al., 2008; Murphy et al., 2008; Papert, 1993; Perkins et al., 1986)
Just in Case	Unnecessary changes such as fixing brackets or additional parentheses. May not always cause a problem	(S. Fitzgerald et al., 2010; Murphy et al., 2008)
Gathering Information/ Clues	Using information from the program output, system messages, and diagnostic print statements about where the bug is	(Carver & Risinger, 1987; Jeffries, 1982; Murphy et al., 2008)
Simple Mapping	Error message points to where the buggy behavior was produced from this output.	(Katz & Anderson, 1987; Klahr & Carver, 1988; Murphy et al., 2008; Romero et al., 2007)

Trial & Error / Editing & Running	Attempting to solve error with different tactics may not have worked and you learn from your mistakes and try again	(Gugerty & Olson, 1986; McCartney et al., 2007; Perkins et al., 1986; Yen et al., 2012)
Reflective (Be persistent, persevere, believe in yourself)	Stay motivated, don't stop trying, persevere, believe in oneself by remembering past successes when debugging	(Deliema et al., 2019; McCartney et al., 2007; O'Dell, 2017; Perkins et al., 1986)
Coming back to the problem later	Moving to another part of the code to return later	(Sue Fitzgerald et al., 2005; McCartney et al., 2007; Perkins et al., 1986)
Switching Gears & Taking Breaks	When you get stuck, switching between tasks or talking a break to return later with fresh mind	(O'Dell, 2017)
"Sweat & Blood Approach"	Sitting down and struggling with the code to effectively	(Gould & Drongowski, 1974)

	debug. Required discipline, motivation, and concentration	
Greedy Search Strategy	Works in stages, consider one input element at a time, at each stage deciding whether an input is part of the optimal solution	(Robert Charles Metzger, 2004)
"Ease into it" Strategy	Avoid relatively difficult sections of the code	(Gould, 1975; Gould & Drongowski, 1974)
Work Around Problem	Worked around a problem rather than facing it by replacing code they did not understand with completely new code	(S. Fitzgerald et al., 2010; Murphy et al., 2008)
Rewriting Sections of Code or Start Over	Solving the problem by avoiding understanding the code and fixing the bug	(Sue Fitzgerald et al., 2008, 2005; Perkins et al., 1986)
Consider Alternatives (Novelty)	Using alternative approaches to problems or thinking of	(Deliema et al., 2019; S. Fitzgerald et al., 2010;

	different ways to solve the problem.	Murphy et al., 2008; Vessey, 1985)
Focused Search Strategy	Focus only on the appropriate section of code where the bug is or may be near	(Carver & Risinger, 1987)
Divide & Conquer of Breaking into smaller parts or inserting breakpoints	Breaking the program into chunks for closer examination	(Araki et al., 1991; Chen et al., 2017; Falahah et al., 2015; McCartney et al., 2007; Perkins et al., 1986)
Elimination	Removing specific areas not essential for debugging the error in the program	(Sue Fitzgerald et al., 2005)
Slicing	Breaking apart the larger program into smaller more manageable pieces	(Badiozamany & Wang, 2010; Grigoreanu et al., 2009; Robert Charles Metzger, 2004; Weiser, 1982)

Isolating the Problem	By commenting out or altering the code to isolate the bug	(S. Fitzgerald et al., 2010; Sue Fitzgerald et al., 2008; Murphy et al., 2008)
Using Resources/Tools	Using resources such as debuggers, internet, blogs and programming manuals More beneficial for programmers with advanced programming knowledge who can comprehend these resources	(Araki et al., 1991; Badiozamany & Wang, 2010; Deliema et al., 2019; Falahah et al., 2015; S. Fitzgerald et al., 2010; Sue Fitzgerald et al., 2008; McCartney et al., 2007; Murphy et al., 2008; O'Dell, 2017; Robert Charles Metzger, 2004)
Asking for Help from teachers or peers (Social support)	Seeking social support and advice from external sources	(Badiozamany & Wang, 2010; Deliema et al., 2019; S. Fitzgerald et al., 2010; Grigoreanu et al., 2009; McCartney et al., 2007)
Running the Program	Run the full program to reason backwards from the output to the potential problem	(Badiozamany & Wang, 2010; Gugerty & Olson, 1986)

Thoroughness	After deciding what is causing the error and how to fix it going back to check that it is correct and produce the anticipated outcome. Checking the fixes, you made.	(Carver & Risinger, 1987; Sue Fitzgerald et al., 2005)
Testing	Difficult to observe because coders simply test the program by running it with no particular goal for the outcome. Can be used as a working progress through debugging	(S. Fitzgerald et al., 2010; Grigoreanu et al., 2009; McCauley et al., 2008; Murphy et al., 2008)
"Means-End" Problem Solving Strategy	Find difference between current and the desired situation and then act to reduce it	(Gould & Drongowski, 1974)
Pattern Matching / Recognition	When code doesn't 'look right' Can be used by novice programmers based on code	(S. Fitzgerald et al., 2010; Sue Fitzgerald et al., 2008, 2005; McCartney et al., 2007;

	they’ve seen before. However, Experts apply higher level reasoning for this application	Murphy et al., 2008; Wiedenbeck, 1985)
Grouping	Looking for similarities or differences in the problem	(Sue Fitzgerald et al., 2005)
Thinking	Processing about what to do or speculating on the possible cause of bugs. This is a mental process of focusing on the problem while also knowing when to take a break. Can be seen during interviews and when the coder has to explain their process	(Deliema et al., 2019; S. Fitzgerald et al., 2010)
Making Connections	Relating to something you already know or a real-world experience to use previous experience/ knowledge	(Deliema et al., 2019; Grigoreanu et al., 2009; McCartney et al., 2007)

Guessing	No reasoning, attempting without sufficient evidence	(Sue Fitzgerald et al., 2005; O'Dell, 2017)
Intuition	Having a feeling without rational and it can be an effective for debugging but it requires extensive experience to be successful	(O'Dell, 2017)
Strategizing	Imagining how you would write the code or what you would need to do	(Sue Fitzgerald et al., 2005)
Posing Questions	Explicitly questioning (what is this? What is the program doing?)	(Sue Fitzgerald et al., 2005)
Depth-First	Create an initial hypothesis for bug location to evaluate fewer portions of the code	(Murphy et al., 2008; Robert Charles Metzger, 2004; Vessey, 1985)

Backward Reasoning	Working backwards from the output The search starts from the incorrect behavior of the program	(Badiozamany & Wang, 2010; Sue Fitzgerald et al., 2008; Gould, 1975; Grigoreanu et al., 2009; Katz & Anderson, 1987; McCauley et al., 2008; Murphy et al., 2008; Romero et al., 2007; Weiser, 1982; Yen et al., 2012)
Top-Down approach	Begin reading the program till the end to find the error	(Alqadi & Maletic, 2017; Chen et al., 2017; Grigoreanu et al., 2009; Lauesen, 1979; Yen et al., 2012)
Casual Reasoning	Look at the information from the code and reasoning about what might be causing the bug	(Carver & Risinger, 1987; Sue Fitzgerald et al., 2008; Gugerty & Olson, 1986; Katz & Anderson, 1987; Murphy et al., 2008; Romero et al., 2007)
Backtracking Approach	Work backwards from the location where the error is	(Falahah et al., 2015)

	manifesting to determine why the it is happening	
Breadth-First	Unfolding the structure of the program by exploring each level of the control-flow hierarchy.	(Murphy et al., 2008; Robert Charles Metzger, 2004; Vessey, 1985)
Forward Reasoning	Searching from the written code by reading the code sequentially or in the order it’ll be executed.	(Sue Fitzgerald et al., 2008; Gould, 1975; Grigoreanu et al., 2009; Katz & Anderson, 1987; McCauley et al., 2008; Murphy et al., 2008; Romero et al., 2007; Yen et al., 2012)
Bottom-up Approach	Narrow down the scope of the error by starting from the last module of the execution flow.	(Chen et al., 2017; Lauesen, 1979)
Following Execution	Viewing the execution of the program in steps while make attention switches between the code and the available output	(Romero et al., 2007)
	Browsing/reading the code to build up a more complete	

Comprehending Program and Code to Gain Domain Knowledge	picture of the program while reading the code. Understanding the program at many levels in order to understand the role of each individual part In order to debug some else's code, you have to understand what it is doing and what it is supposed to do	(S. Fitzgerald et al., 2010; Gould & Drongowski, 1974; Grigoreanu et al., 2009; Gugerty & Olson, 1986; Jeffries, 1982; Katz & Anderson, 1987; Kessler & Anderson, 1986; Murphy et al., 2008; Romero et al., 2007)
Self-Terminating Brute Force Search Strategy	Reading and simulating every instruction until the bug is located and then disregards the rest	(Carver & Risinger, 1987)
Brute-Force Search Strategy	Reading and simulating every instruction	(Carver & Risinger, 1987)

Appendix B: Debugging Mission Protocol

Introduction (5-10 min)

- Introductions of parent, child, and researcher
- Explain the purpose of the present study
- Parent and child sign consent/assent forms (recording start upon approval)

Demographic Survey - (10 min):

- Parents will focus on filling out the demographic survey and the Researcher will begin explaining the debugging missions to the child

Explanation of debugging missions (5 min):

- Explain: What are the debugging missions?
- There are 4 missions- once you've completed one mission you can move on to the next, but if you don't complete a mission that's okay.

Mission 1: *KIBO:* Let's go to the zoo! **(syntax error in code)**

- Read prompt with child
- Researcher is allowed to provide prompts and cues as well as asks questions when necessary

Mission 2: *KIBO:* Let's go to the beach! **(semantic error in code)**

- Read prompt with child
- Researcher is allowed to provide prompts and cues as well as asks questions when necessary

Mission 3: *KIBO:* Let's get KIBO to school! **(systemic error in code)**

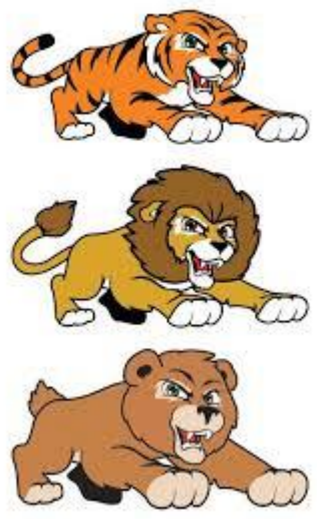
- Read prompt with child
- Researcher is allowed to provide prompts and cues as well as asks questions when necessary

Mission 4: *KIBO:* KIBO is going on a trip! **(correspondence error in code)**

- Read prompt with child
- Researcher is allowed to provide prompts and cues as well as asks questions when necessary
- Once completed, let the child know they're done!

Appendix C: Debugging Missions

KIBO TAKES ON THE ZOO!



KIBO wants to go to the ZOO today!

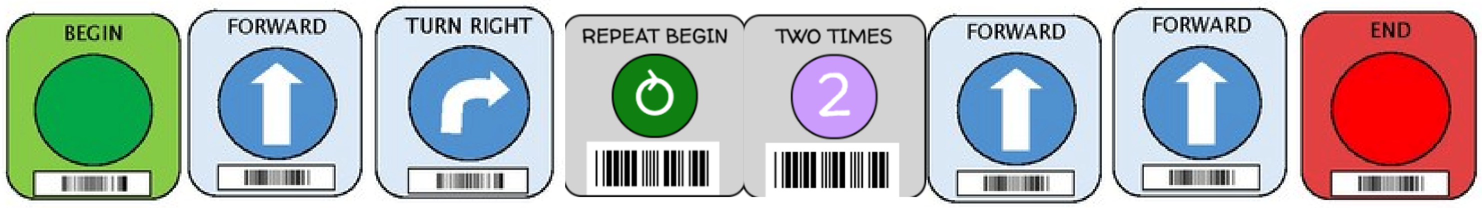
KIBO is meeting other KIBO's at the ZOO to go see some cool animals!

They hope to see lions and tigers and bears OH MY!

KIBO was ready to go to the ZOO, but KIBO keeps getting lost….

A friend scanned KIBO with the code below

Scan KIBO with this code and see why KIBO is not making it to the ZOO!



Let's get KIBO to the ZOO!

BEACH DAYS ARE THE BEST DAYS!

KIBO and their friends want to go to the beach!

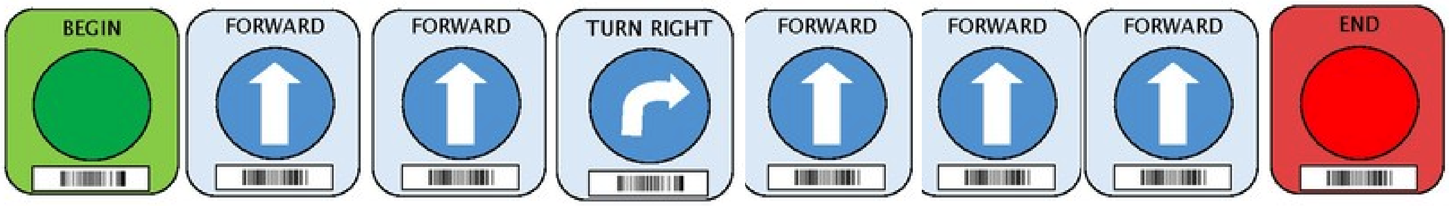
Robots like to build sandcastles and get a tan just like us.

The KIBO’s packed up their shovels, pails, beach chairs, and sunscreen!

A friend scanned a code to get them to the beach, but it did not work.

The KIBO’s keep missing the beach and they don’t want to miss out on their perfect beach day!

This is the code our friend scanned KIBO with to get them to the beach.



Let’s help KIBO get to the beach!

KIBO IS ON A MISSION TO GET HOME!

KIBO needs help getting home!

KIBO was visiting a friend and it’s time for KIBO to get back to KIBO’s family.

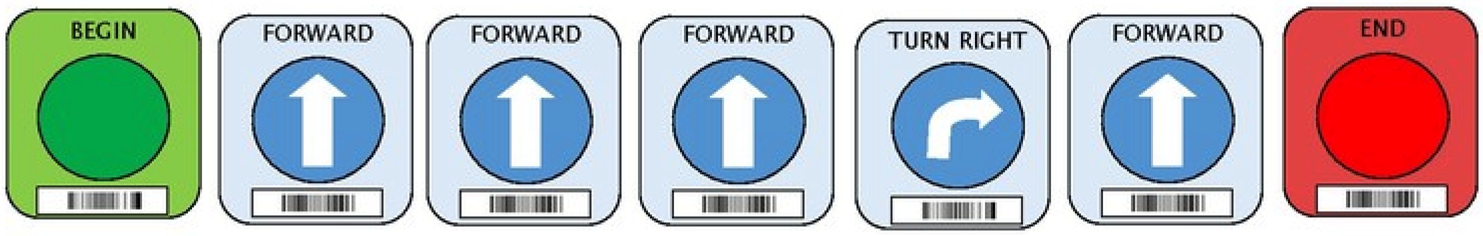
A friend tried to help KIBO by scanning a code to get KIBO home, but it didn’t work.

OHHH NO!

Let's help KIBO get home!

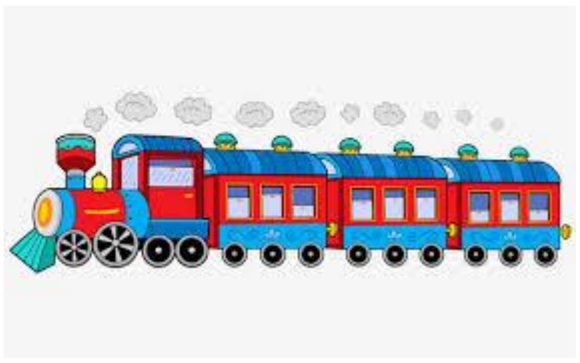
This is the code that our friend used to try and get KIBO home.

Scan KIBO with this code and see why KIBO is not making it home!



How can we use this code to get KIBO home?

KIBO IS LATE! LATE FOR A VERY IMPORTANT DATE

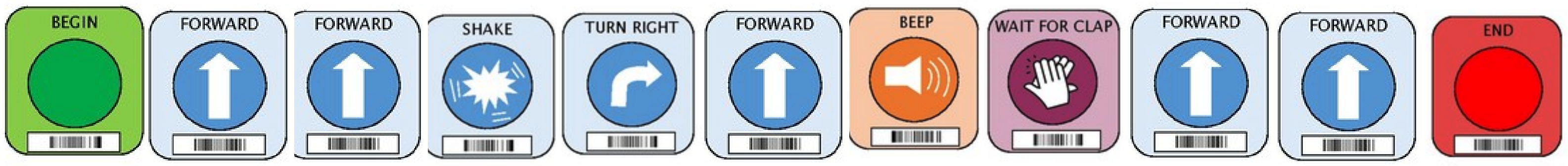


KIBO is trying to get to their train to go on a trip!

The train leaves very soon and KIBO can't miss it or KIBO will be sad.

A friend is trying to get them to the train, but it is not working, and they do not have much time.

Our friend used this code to try and get KIBO to the train, but it would take KIBO too long to get there.



How can we fix this code to get KIBO to the train?

WHAT TIME IS IT? IT IS TIME FOR SCHOOL!

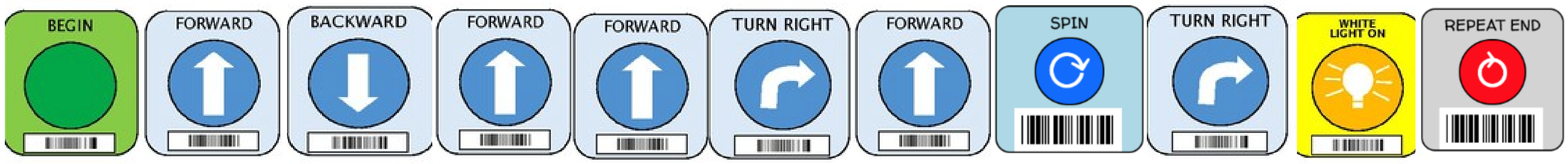
KIBO has to go to school just like us.

They go to a special robot school to learn how to read code!

It is getting close to school time and KIBO has their backpack ready to go with all of their books and pencils.

KIBO can’t be late for school or they will miss important information!!!

We need to get KIBO to school on time, but the code is taking way too long to get them there.



Can you get KIBO to school on time!

Appendix D: Participant Portrait by Mission

	P0
Mission 1	Methodical thinking, identifies error but does not know how to apply any of it, connects or makes association to a prior early learned concept 8:21 "do re mi fa so la ti do", 9:48-11:42 a great example of how participant approaches thinking (thinking, makes a guess then thinks, provides rationale for why, then does the action for guess, identifies the problem at the system level, encourages KIBO, also identifies where his thinking was wrong, "it must be all KIBO's not just this KIBO" - system level thinking, *concept vs. application*
Mission 2	correctly identifies the bug, methodical approach by changing one block at a time, facial expression when he arrives at the end destination
Mission 3	encourages KIBO, responds to KIBO feedback, gets frustrated when KIBO executes code by accident, taps the code blocks while saying what KIBO is actually doing, quote - P0 is thinking and makes "hmmmm" noise (C1 at 1:45 has visual of P0 thinking he has the answer), guesses the error is a scanning problem, says code out loud as he scans - only certain blocks within the code, P0 throw arms up and says "What on earth is KIBO doing" (let's KIBO finish its code) (C2 visual of arms thrown at 22:16), thinks and processes, "he should now know it" expecting the robot to know how to proceed, explains reasoning for guesses, checks hardware, finds error, P0 pushes ▵ to begin the program and as it starts moving says "there" (as a sigh of relief for figuring out the issue - (big smile) (C2 has visual facial expressions of P0 response to correcting the error at 27:46), then says "I completed it!" channels frustration to motivation
Mission 4	excited about mission theme, makes guess for what KIBO might need, but begins to scan the original code first, reads code in order to clap when KIBO needs it, claps several times in an attempt to get KIBO to hear, encourages - KIBO along the way saying "come on man!", has "aha moment" when realizes KIBO needs and ear, sigh of relief when solving problem - stating "there" again, genders KIBO saying "he made it!"

	P1
Mission 1	asks clarifying questions, concept vs. application - participant knows what the repeat block does but not how to use it, uses his body to pretend he is KIBO to visualize or understand the code, whenever he guesses he provides rationale and then checks his guesses, he narrows it down to two options for what it might be and picks one to go with, guess and check process - he guesses something, takes action based on guess, then checks to verify and adjusts accordingly, seeks feedback from researcher looks for guidance, responds to feedback provided by error message "beep" aka KIBO feedback, after completing a mission he is able to explain what the error was - he understands (guess, provides rational, KIBO error beep, processes then adjusts - this is the **methodical** process)
Mission 2	crawls code out - walks it out, identifies bug, takes action to correct bug, checks, makes second guess, identifies problem with the guess, reverts back. Responds to KIBO feedback, says "he made it!" when KIBO arrives at destination/mission end
Mission 3	tests by crawling out code - walks through it, notices KIBO output is not what it is supposed to do, participant responds by saying "nope" in response to what KIBO is doing, looks between code and end destination multiple times, knows he needs to change something, thinks the code is correct, breaks code into 3 sections, responds to KIBO feedback, physical movement of arms thrown up and says "no" time 26:15, looks at hardware, identifies error, corrects error, smiles when KIBO makes it, is able to explain error
Mission 4	crawls out code, responds to KIBO feedback when scanning, says "nope" when KIBO doesn't make it, throws arms up as a sign of confusion and frustration, "I don't know", identifies error and seeks to correct it, "I think"… "I scanned all of them I know", claps after KIBO beep, says "KIBO made it!" and throws hand up in excitement

	P2
Mission 1	uses the word "maybe" constantly (this could be a signal of uncertainty), genders robot as "he", displays methodical thinking by moving one thing at a time, ignores some KIBO feedback, makes an output adjustment to refer back (changes code, scans KIBO, lets KIBO run, and it malfunctions, then reverts back until she got a new KIBO and tries to start debugging from prior KIBO, focuses on the unknown part of the problem to remove it, #block connection if the blocks don't connect together the participant then stops doing what she is doing and trying guessing something else, miss/improperly scans blocks constantly, thoughts (needs prompts and guidance to express her thoughts, she is emotionally regulated and does not struggle or get frustrated despite spending 20 minutes on this mission)
Mission 2	re-scan, identifies KIBO going the wrong direction, points the rest of the way KIBO needs to go to get to end destination, understood the what the code was supposed to do and was aware that it didn't do that, mis scans code, laughs when KIBO hits the wall, separates the code into segments to focus on things, rescans, ignores KIBO feedback and error message, genders KIBO as "he", facial expression of happiness at the end of mission
Mission 3	relates previous mission (M1) where KIBO almost got home, mis-scanning blocks, "No KIBO that's not what it was", walks out code, thinks it's a problem with the code then realizes its KIBO, no idea=scan again, creates new code and tests by walking out before scanning KIBO, thinks, revisits the idea that it's a problem with the code, checks hardware, identifies error, seeks feedback from researcher, wants to try exact code from mission paper, methodical approach-one block at a time, **#block connection=tactic**, checks throughout the process, happiness
Mission 4	excited/fidgeting and playing with her hair, asks clarifying questions, mis-scan, checks hardware, identifies missing KIBO part, removes unnecessary blocks from original mission code (shake, beep, clap), using finger to point path, **#block connection**, methodical approach to moving blocks around in the code, place blocks on the floor laying out the path (visualization), "this is definitely going to work".

	P3
Mission 1	gets **distracted** easily, runs program (push triangle) while looking at research and then keeps persistently hitting triangle in a way that is obviously showing the he is frustrated/upset, reflects on another time when KIBO wasn't working, hits himself in the face for scanning forwards - clearly emotionally unregulated
Mission 2	runs all over the room, begins mission by counting the number of blocks needed to get KIBO to destination, #block connection (ex. Blocks don't connect, tries to shove blocks together, then when fails participant separates them), makes 3 block codes (ex. begin, movement, end) when participant scans they only scan the middle block each time - emotion is noticeable it's a roller coaster frustrated, doesn't want to follow directions when putting KIBO on the start block, draws on whiteboard to show where KIBO is and where the end destination is
Mission 3	distracted, jumps when about to start mission, pushes KIBO body to the end destination (anger/frustration), walks around room, tries to run program multiple times but then just pushes KIBO body, Frustrated enough to hit KIBO body with fist=approach is to hit robots when frustrated at it, mentions other frustration, scans blocks in a tower form, attempts to add sensors, pushes robot with foot, compares KIBO hardware with another KIBO, wants to move to next mission, and finally just wants to stop missions
Mission 4	N/A - gave up

	P4
Mission 1	scans each block individually, wants to add all of the sensors, tries to run program consistently, crawls out code - walk through it to visualize, expresses large boughs of emotion (excitement) when the mission is completed
Mission 2	laughs at mission, wants to add sensors, individually scans code, identifies bug, puts the code block together the same way the mission paper displays it but switches the turn, begins scanning one block at a time, excitement for KIBO before even finishing mission
Mission 3	adds sensors, "hey KIBO did you get that?" - verbalizes KIBO did not do what the program was supposed to do (moved close to KIBO to speak to KIBO for understanding), runs program, picks KIBO up and places it at the end destination, rescans using one block at a time, checks hardware, asks clarifying questions, identifies problem and fixes it, excitement and clapping
Mission 4	adds sensors, scans one block at a time, asks clarifying questions, creates new code, still scanning one block at a time, adjusts from KIBO feedback, excitement when KIBO arrives at destination

	P5
Mission 1	listens to mission, changes code at first attempt, tries to run program multiple times, adjusts code based on feedback from KIBO, adds one block then runs program, wants to scan KIBO from where last code output/feedback/ended up then makes a new code, runs it, then solves mission
Mission 2	looks closely at mission for over 10 seconds then tries the code, wants the code from where KIBO's last code left him, keeps pushing triangle to move KIBO all around, picks KIBO up and moves it to the end destination, walks code out, makes own code and does mission in pieces, says maybe a lot, adjusts based on feedback
Mission 3	doesn't try original code instead creates new code, laughs while KIBO does code, runs program again, checks hardware, verbalizes finding the error "no green light" but responds "I don't know" when asked what to do next, corrects error but not able to code KIBO to end destination, codes KIBO then codes KIBO from that spot to get to the end destination
Mission 4	runs program, asks for mission paper to be read again because participant did not listen to instructions at the start of the mission, scans original code-"this will work", claps but KIBO does not respond - participant looks at research and says "is there something wrong" - participant moves closer to KIBO to clap again, checks hardware, re-runs program, claps and KIBO does not respond so participant guesses to check the batteries, thinks something is broken, finally discovers error and adds ear, jumps up and down with excitement when KIBO gets to end destination

	P6
Mission 1	tries to run program constantly, checks hardware, thinks he needs a new KIBO - one system vs all system error - he thinks it's a system error even though it is not, guess KIBO needs sensors even though he knows KIBO doesn't need them but might want them, **#block connections**, understands concept of the repeat block, crawls through code to visualize it and walk through the path that KIBO would travel to try to reason through the problem, methodical thinking (ex. takes it a block at a time), he is **excited** when KIBO gets to destination,
Mission 2	listens to mission, runs program, "it's going the wrong way!" grabs KIBO in middle of code, uses the feedback to take turn block out and put the other turn block in then rescans code - smiles while saying "that was easy"
Mission 3	switches attention between listening and scanning, responds to KIBO output with "what the…what the…", rescans, breaks code apart to make new code, makes smaller pieces of code to check if spinning is still happening, but forgets an end block often while making small codes, noticed KIBO is doing the opposite of the code and approaches it by giving KIBO the opposite task, identifies error, corrects error, checks using smaller pieces of the code to make sure it works - checking, runs program and adjusts from KIBO feedback - methodically by only removing one block, solves error and dances around while transitioning to next mission
Mission 4	runs program, KIBO beeps as KIBO is coded to do so but participant is confused about what they should do or what it means, so they add blocks at the end of the code, scans new code and runs program, checks hardware, notices robot stops after beep, when asked about the KIBO stopping at the beep participants approach is verbalized as ignore it that's the best thing to do and that the beep is annoying, adds more blocks to the end of the code, checks hardware, guesses batteries could be the problem, realizes the need to clap and wants to take the wait for clap block out, removes the wait for clap block, makes loud noise "ghhhh" when KIBO's output is far away, adjusts code based on feedback, solved mission and could identify that the clap was the problem

	P7
Mission 1	seeks feedback from researcher, asks clarifying questions, makes multiple guesses at one time (batteries are bad, wires tangled his motors - could be from experience or age level), concept vs. application (ex. Understands what the repeat block does but not the application), process: problem - solution - rationale, adjusts her strategy based on feedback (ex. "we shouldn't try scanning it, it will not work"), uses KIBO feedback to make adjustments, excited when KIBO gets to the end, addresses KIBO in humanoid way (ex. "he doesn't feel well"), then displays understanding as to difference between human and robot language, then she **encourages** the robot like you would encourage a human
Mission 2	guesses then explains guess which results in identifying the bug, gets distracted and forgets to scan KIBO, notices KIBO feedback, seeks feedback from researcher, wants to check that the left block will turn left, happy when mission is solved
Mission 3	runs program, "hmmm" - noise as participant is confused, rerun code, seeking feedback from researcher, genders KIBO in to "he" male gender as participant says "he's being a weirdo" - is mean to robot, "I told him to do this" - participant is rationalizing why she is confused that KIBO does not work properly, try again, makes possible guess but does not follow up with it, negatively references KIBO "I'm stopping the dumb KIBO", relates this problem to another mission problem (cross-association), identifies KIBO output as what the participant believes the KIBO is doing and draws it out on the floor with participants finger, makes guess about the error, gets distracted and makes prompting back, takes action on the guess, provides explanation for the guess, checks hardware again, makes more guesses but does not follow through with any of them, attempts to adjust error by physically move KIBO while program is running, participant is excited and encourages KIBO when it gets to the end destination by saying "KIBO made it home!...KIBO is a good KIBO", could explain error after little prompting
Mission 4	asks clarifying questions, distracted, understands code to know to clap, makes a guess, takes action on the guess, checks, solves mission and participant looks **surprised** as she recognized she did not solve the error, could explain error after, then expresses **excitement**

	P8
Mission 1	scans code in an odd way, checks hardware (ex. Sensors, hardware, wheels), references mission on paper for clarification, tries to run program many times and gets **frustrated**, gets rid of code to make new one, crawls out then pieces out blocks together (walks out code), when KIBO moves participants says "Yay! I got it!", "I'm just going to push him to the zoo because he will never get there" then says "I want to move on" at time 27:09 minutes into mission
Mission 2	notices error when scanning code, realizes she has to change a block, #block connection, solves error to complete mission and runs KIBO one more time just for fun
Mission 3	interrupts mission being read, walks code out already saying "it doesn't work", asks questions, removes blocks, crawls code out, and revises code from walking it out, "huh" confused when KIBO spins, walks it out again, adds sensors, frustrated when KIBO is still spinning, tries a different KIBO body, relates an injury she has on her body (participants body) to "KIBO's booboo", checks hardware and compares it with other KIBO, finds error and solves it, runs KIBO, uses foot to push KIBO to end destination
Mission 4	before starting mission participant steps on KIBO, understood program to know to clap, runs program again, asks researcher to read mission paper again, wants to move start marker, participant moves KIBO's body while saying the blocks she thinks it needs to use, adds recorder sensor, when it doesn't work, blames not scanning the code, runs program and claps but nothing happens, from KIBO feedback guesses the block should be removed, scans new code but mis scans, wants researcher to scan, KIBO makes it to end destination

	P9
Mission 1	says "this is hard" at time 14:17 minutes into mission, says "I never give up" at 15:18 minutes into mission, constantly adds new errors/bugs (ex. claps, sounds, light) when unnecessary, asks lots of questions, decides to make her own code instead of using the mission code because she did not think they looked the same, she references the paper mission picture for needing sensors on KIBO, gets distracted and is very disorganized, to solve mission she has to chunk it out - wants to return to prior output - referring back to prior point
Mission 2	adds blocks randomly, asks questions, lots of distractions, stories, and jokes, watches KIBO hit wall when still doing code, has no idea that didn't work, responds to KIBO output and then removes blocks accordingly, KIBO hits wall again, then participant takes more blocks away, "I got this guys!" and "that was easy" - determination
Mission 3	asks many questions because participant is overly excited before mission starts, before mission is read: "this is going to be hard" then realizes it may not be too hard, interrupts multiple times while researcher is reading the mission, identified school on mission paper and end destination are not the same, looks at researcher for feedback, thinks picture on blocks are not same as the barcode (forward and spin blocks have pictures swapped), uses finger to poke along floor saying blocks needed (large amount of blocks), creates new code until turn then goes back to figure out the rest of the blocks needed, guesses batteries are a problem, talks w/ researcher, guesses wires are problem, guesses screw not all the way in as problem, idea to use spin block to make KIBO move forward, needs more spin blocks so tries to add repeat loop even though little knowledge of what a repeat block does, uses 1 spin to see if it works and it does, blames barcodes as problems, frustrated explaining the sticker is for humans not KIBO "we don't know what is going on", tries another KIBO to check spin/forward , guess problem with KIBO, guesses random error that could be wrong with hardware, "if you change it as it works then that was the problem", tries to use repeat block concept vs. application (some knowledge of repeat blocks) attempts nested loops, does code in pieces of smaller code and adjusts based on feedback, solves mission by doing changes of the code using spin blocks as forwards (never identifies wheels as error but solves mission)
Mission 4	N/A ran out of time

	P10
Mission 1	seeks feedback from researcher, identifies the error but no change in approach, narrows down problem to the code, tries running the program multiple times which results in her getting frustrated, taps on code block pointing out the color or function of the block, when participant doesn't know the block she guesses it's function, genders KIBO as "he", ends up removing blocks that she does not understand, mis scans which ultimately leads to solving the error, positive emotion when KIBO arrives at destination/end of mission
Mission 2	makes "ehh" noise/grunt when trying to scan KIBO, identifies KIBO is going the wrong way, verbally states the way KIBO is going then what the correct way KIBO should go and taps the blocks while doing so, #block connection, asks for help with blocks
Mission 3	shocked facial expression at KIBO's output, uses hands to adjust KIBO body in the correct direction, rescanning, ignores KIBO feedback of error message, noises and physical movement demonstrate frustration, taps on blocks saying what KIBO should be doing, sounds frustrated when KIBO is not listening, tries to run program multiple times, getting more aggressive with each attempt, checks hardware, giggles and makes jokes about KIBO, asks clarifying questions, runs program aggressively pushing KIBO start button triangle and giggles, adds unnecessary blocks and sensors, gets frustrated when those blocks and sensors when KIBO doesn't do them even though she didn't scan a new code to use them, slams light bulb repeatedly on and off KIBO, notices the spark of electricity and light bulb which makes her excited so she slows down and gently lifts light bulb off to show the light, expresses not knowing why KIBO isn't working because she tested the code and doesn't know what to do next: strategy 'talks about code and doesn't know what to do next', says I don't know and checks hardware, identifies error, shows excitement, several references to excitement as the mission is solved
Mission 4	before mission starts participant is experiencing a lot of emotions, overly excited and giggling, potentially unregulated emotions in response to solving last mission, runs program slight aggressively, moving paper mission out of the way of coding blocks, noises expressing over excitement and overconfidence, makes noises that she is confused, checks hardware, tries to run program, says maybe and makes guesses, expresses excitement when KIBO triangle light is flashing green, seeks feedback from researcher, gets excited and giggles when KIBO runs program but as soon as KIBO beeps and stops moving fast emotion change, makes jokes and giggles, discovers error, fixes error, attempts to run program but had mis scanned blocks, frustrated leads to "I don't know" and walks away for water break and returns, physically pushes KIBO body and giggles, gets distracted, tries running program again aggressively attempts moving sensor around KIBO body, emotionally unregulated as she gets distracted and knocks things over using KIBO, readjusts and rescans code, KIBO gets to the end destination, participant could not identify how to fix the code but with direction describes the error and then jumps with excitement

	P11
Mission 1	genders KIBO as "he", uncertainty by using the word "maybe" constantly, guesses it's the batteries 3 times without following up or checking/verification, asks clarifying questions, displays emotions using body movement, gets distracted by sticky notes in room, adds unnecessary blocks, #block connection decides tactic, concept vs. application (ex. thinks the repeat block is a twist but applies it correctly), ignores KIBO feedback - does not know what it means, attempts to run program multiple times, re-scan and successful mission completion
Mission 2	ask researcher to cover blocks while research is still reading the mission, identifies error and then corrects it, #block connection, says maybe a lot and guesses, physical movement - frustration as he doesn't know why KIBO is malfunctioning, says "I don't know" a lot, watches KIBO even though he goes past the destination, wants to move end destination, distracted, moves end destination block in front of KIBO, guesses verbally in an escalated voice when saying "I WONDER IF we can add some ears", says "he's at the beach" (KIBO isn't at the beach), frustration (ex. Researcher asks about what blocks would they be and P11 responds "I don't know" as P11 walks over the bucket (throw arm up) and says under breath "I don't know what kind of blocks" (throw arm up) (C2 at 24:28 visual of arm thrown)), Pushing the KIBO in a frustrated/ aggressive way, says "done" and has been turning the wheels on the robot
Mission 3	distracted before mission even starts, "KIBO is having a really busy day", says "I don't know" and shrugs, uses hand to push KIBO to the end destination, identifies that it is something wrong with KIBO but says "maybe", checks hardware, guesses it's the batteries, compares batteries with all other KIBO's, "I don't know", gets distracted, "I don't know", wants to stop this mission and go on to the next one
Mission 4	runs program, didn't look at or understand the code as he thought the KIBO beep meant it was done, adds blocks, gets distracted, tells a story about KIBO getting to the end destinations, lays down and says "I don't know", tries to make guess gets distracted mid thought, says "maybe" and makes guess that rescan blocks will solve the problem and does not look at the code, distracted, identifies the need to clap but when it doesn't fix problem leads to unproductive attempts at running KIBO, uses body as barrier in front of KIBO so it hits him, rolls over and says "let's just skip this one"

References

Ahmadzadeh, M., Elliman, D., & Higgins, C. (2005). An Analysis of Patterns of Debugging Among Novice Computer Science Students. *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 84–88. https://doi.org/10.1145/1067445.1067472

Alqadi, B. S., & Maletic, J. I. (2017). An Empirical Study of Debugging Patterns Among Novices Programmers. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 15–20. https://doi.org/10.1145/3017680.3017761

Araki, K., Furukawa, Z., & Cheng, J. (1991). A general framework for debugging. *IEEE Software*, *8*(3), 14–20. https://doi.org/10.1109/52.88939

Badiozamany, S., & Wang, H. (2010). *Debugging: The Difference between Novices and Experts*.

Bers, M. U. (2018). Coding, playgrounds and literacy in early childhood education: The development of KIBO robotics and ScratchJr. *2018 IEEE Global Engineering Education Conference (EDUCON)*, 2094–2102. https://doi.org/10.1109/EDUCON.2018.8363498

Carver, S., & Risinger, S. (1987). *Improving children's debugging skills*. 147–171.

Chen, W.-K., Liu, C.-H., & Huang, H.-M. (2017). *Software Debugging Patterns for Novice Programmers*. 17.

Deliema, D., Dahn, M., Flood, V., Asuncion, A., Abrahamson, D., Enyedy, N., & Steen, F. (2019). *Debugging as a Context for Fostering Reflection on Critical Thinking and Emotion* (pp. 209–228). https://doi.org/10.4324/9780429323058-13

Denham, Susanne A. (1998). *Emotional development in young children* (Guilford series on social and emotional development). New York: Guilford Press.

Denham, S. A., & Bouril, B. (1994). Preschoolers' affect and cognition about challenging peer situations. *Child Study Journal*, *24*(1), 1.

Ekman, P. (1989). "The argument and evidence about universals in facial expressions of emotion". In H. Wagner & A Manstead (ed.). *Handbook of social psychophysiology*. Chichester, England: Wiley. pp. 143–64.

Ekman P (2003) Changing what we become emotional about. In: Ekman P (ed) Emotions revealed. Times Book (Henry Holt and company) LLC, New York, pp 67–68

Falahah, Suwardi, I. S., & Surendro, K. (2015). General pattern identification of debugging system. *2015 International Conference on Information Communication Technology and Systems (ICTS)*, 67–72. https://doi.org/10.1109/ICTS.2015.7379873

Fitzgerald, S., McCauley, R., Hanks, B., Murphy, L., Simon, B., & Zander, C. (2010). Debugging From the Student Perspective. *IEEE Transactions on Education*, *53*(3), 390–396. https://doi.org/10.1109/TE.2009.2025266

Fitzgerald, Sue, Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: Finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education*, *18*(2), 93–116. https://doi.org/10.1080/08993400802114508

Fitzgerald, Sue, Simon, B., & Thomas, L. (2005). Strategies That Students Use to Trace Code: An Analysis Based in Grounded Theory. *Proceedings of the First International Workshop on Computing Education Research*, 69–80. https://doi.org/10.1145/1089786.1089793

Gough, C., Ledermann, J., & Elms, K. (1994). *Interpretive Debugging of Optimised*.

Gould, J. D. (1975). Some psychological evidence on how people debug computer programs. *International Journal of Man-Machine Studies*, *7*(2), 151–182. https://doi.org/10.1016/S0020-7373(75)80005-8

Gould, J. D., & Drongowski, P. (1974). An Exploratory Study of Computer Program Debugging. *Human Factors*, *16*(3), 258–277. https://doi.org/10.1177/001872087401600308

Graziano, P. A., Reavis, R. D., Keane, S. P., & Calkins, S. D. (2007). The Role of Emotion Regulation and Children's Early Academic Success. *Journal of school psychology*, *45*(1), 3–19. https://doi.org/10.1016/j.jsp.2006.09.002

Grigoreanu, V., Brundage, J., Bahna, E., Burnett, M., ElRif, P., & Snover, J. (2009). Males' and Females' Script Debugging Strategies. In V. Pipek, M. B. Rosson, B. de Ruyter, & V. Wulf (Eds.), *End-User Development* (pp. 205–224). Springer Berlin Heidelberg.

Grover, S., Cooper, S., & Pea, R. (2014). Assessing Computational Learning in K-12. *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, 57–62. https://doi.org/10.1145/2591708.2591713

Gugerty, L., & Olson, G. (1986). Debugging by Skilled and Novice Programmers. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 171–174. https://doi.org/10.1145/22627.22367

Jeffries, R. (1982). *A Comparison of the Debugging Behavior of Expert and Novice Programmers*. 17.

Katz, I. R., & Anderson, J. R. (1987). Debugging: An Analysis of Bug-Location Strategies. *Human–Computer Interaction*, *3*(4), 351–399. https://doi.org/10.1207/s15327051hci0304_2

Kazemian, F., & Howles, T. (2008). *Teaching Challenges—Testing and Debugging Skills for Novice Programmers*.

Kessler, C. M., & Anderson, J. R. (1986). A Model of Novice Debugging in LISP. *Papers Presented at the First Workshop on Empirical Studies of Programmers on Empirical Studies of Programmers*, 198–212. http://dl.acm.org/citation.cfm?id=21842.28895

Klahr, D., & Carver, S. M. (1988). Cognitive objectives in a LOGO debugging curriculum: Instruction, learning, and transfer. *Cognitive Psychology*, *20*(3), 362–404. https://doi.org/10.1016/0010-0285(88)90004-7

Klahr, D., & Robinson, M. (1981). Formal assessment of problem-solving and planning processes in preschool children. *Cognitive Psychology*, *13*(1), 113–148. https://doi.org/10.1016/0010-0285(81)90006-2

Kostelnik, Marjorie J. (2012). *Guiding children's social development and learning* (7th ed.). Belmont, CA: Wadsworth/Cengage Learning.

Lauesen, S. (1979). Debugging techniques. *Software: Practice and Experience*, *9*(1), 51–63. https://doi.org/10.1002/spe.4380090106

Lockwood, J., & Mooney, A. (2017). Computational Thinking in Education: Where does it Fit? A systematic literary review. *ArXiv:1703.07659 [Physics]*. http://arxiv.org/abs/1703.07659

Mayeux, L., & Cillessen, A. H. N. (2003). Development of Social Problem Solving in Early Childhood: Stability, Change, and Associations With Social Competence. *Journal of Genetic Psychology*, *164*(2), 153. https://doi.org/10.1080/00221320309597975

McCartney, R., Eckerdal, A., & Moström, J. E. (2007). *Successful students' strategies for getting unstuck*. 5.

McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: A review of the literature from an educational perspective. *Computer Science Education*, *18*(2), 67–92. https://doi.org/10.1080/08993400802114581

Murphy, L., Lewandowski, G., McCauley, R. A., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: The good, the bad, and the quirky – a qualitative analysis of novices' strategies. *SIGCSE*. https://doi.org/10.1145/1352135.1352191

Nehaniv, C. L., & Dautenhahn, K. (2002). *The Correspondence Problem in Social Learning:– What Does it Mean for Behaviors to Match Anyway*.

O'Dell, D. H. (2017). The Debugging Mindset. *Queue*, *15*(1), 50:71–50:90. https://doi.org/10.1145/3055301.3068754

Papert, S. (1993). *Mindstorms: Children, computers, and powerful ideas* (2nd edition). Basic Books.

Perkins, D. N., Hancock, C., Hobbs, R., Martin, F., & Simmons, R. (1986). Conditions of Learning in Novice Programmers. *Journal of Educational Computing Research*, *2*(1), 37–55. https://doi.org/10.2190/GUJT-JCBJ-Q6QU-Q9PL

Perkins, D. N., & Martin, F. (1986). Fragile Knowledge and Neglected Strategies in Novice Programmers. *Papers Presented at the First Workshop on Empirical Studies of Programmers on Empirical Studies of Programmers*, 213–229. http://dl.acm.org/citation.cfm?id=21842.28896

Raver, C. C., Garner, P. W., & Smith-Donald, R. (2007). The roles of emotion regulation and emotion knowledge for children's academic readiness: Are the links causal? In R. C. Pianta, M. J. Cox, & K. L. Snow (Eds.), *School readiness and the transition to kindergarten in the era of accountability* (p. 121–147). Paul H Brookes Publishing.

Robert Charles Metzger. (2004). *Debugging by Thinking*. Elsevier. https://doi.org/10.1016/B978-1-55558-307-1.X5000-9

Romero, P., du Boulay, B., Cox, R., Lutz, R., & Bryant, S. (2007). Debugging strategies and tactics in a multi-representation software environment. *International Journal of Human-Computer Studies*, *65*(12), 992–1009. https://doi.org/10.1016/j.ijhcs.2007.07.005

Rubey, R. J. "A comparative Evaluation of PL/1". Datamation, December, 1968, 20-25.

Saarni, Carolyn, & Harris, Paul L. (1989). *Children's understanding of emotion* (Cambridge studies in social and emotional development). Cambridge [England] ; New York: Cambridge University Press.

Shaochun Xu, & Rajlich, V. (2004). Cognitive process during program debugging. *Proceedings of the Third IEEE International Conference on Cognitive Informatics, 2004.*, 176–182. https://doi.org/10.1109/COGINF.2004.1327473

Simon, B., Fitzgerald, S., McCauley, R., Haller, S. M., Hamer, J., Hanks, B., Helmick, M. T., Moström, J. E., Sheard, J. I., & Thomas, L. (2007). Debugging assistance for novices: A video repository. *ITiCSE-WGR '07*. https://doi.org/10.1145/1345443.1345437

Simon, H. A., & Newell, A. (1971). *Human problem solving: The state of the theory in 1970.* https://doi.org/10.1037/h0030806

Sullivan, A., Elkin, M., & Bers, M. U. (2015). KIBO robot demo: Engaging young children in programming and engineering. *IDC*. https://doi.org/10.1145/2771839.2771868

Vessey, I. (1985). Expertise in debugging computer programs: A process analysis. *International Journal of Man-Machine Studies*, *23*(5), 459–494. https://doi.org/10.1016/S0020-7373(85)80054-7

Vohs, Kathleen D, Baumeister, Roy F, & Loewenstein, George. (2007). *Do emotions help or hurt decision making? : A hedgefoxian perspective*. New York: Russell Sage Foundation.

Weiser, M. (1982). Programmers Use Slices when Debugging. *Commun. ACM*, *25*(7), 446–452. https://doi.org/10.1145/358557.358577

Wiedenbeck, S. (1985). Novice/expert differences in programming skills. *International Journal of Man-Machine Studies*, *23*(4), 383–390. https://doi.org/10.1016/S0020-7373(85)80041-9

Wing, J. M. (2006). *Computational thinking*.

Yen, C.-Z., Wu, P.-H., & Lin, C.-F. (2012). Analysis of Experts' and Novices' Thinking Process in Program Debugging. In K. C. Li, F. L. Wang, K. S. Yuen, S. K. S. Cheung, & R. Kwan (Eds.), *Engaging Learners Through Emerging Technologies* (pp. 122–134). Springer Berlin Heidelberg.