Defining and Supporting a Debugging Mindset in Computer Engineering Courses

Henry Duwe, Diane T. Rover, Phillip H. Jones, Nicholas D. Fila
Electrical and Computer Engineering
Iowa State University
Ames, IA USA
{duwe, drover, phjones, nfila}@iastate.edu

Mani Mina
Industrial Design and
Electrical and Computer Engineering
Iowa State University
Ames, IA USA
mmina@iastate.edu

***Abstract*—While it is commonly held that debugging is a critical activity for engineers, particularly computer engineers, it is rarely a core component in engineering curriculum. Often it is either considered an innate skill or one to be developed indirectly through coursework. We argue that debugging is more authentically a constellation of mindsets or intrinsic beliefs, values, and dispositions that orient behavior. We define the debugging mindset, describe a series of activities to support the development of such a mindset, and offer evidence of a debugging mindset among students in two computer engineering project courses.**

***Index Terms*—debugging, testing, mindsets, engineering education**

I. Introduction

Engineering educators often do not explicitly emphasize debugging and testing in their classes, and students are often lacking in these competencies. However, developing testing methods and skills for verifying and validating that an implemented system is behaving as specified, and applying effective debugging strategies to identify and correct issues are essential to responsible development of products and technologies. While debugging and testing are often framed as distinct skills that students and engineers may develop, such competencies more authentically represent related mindsets, or intrinsic beliefs, values, and dispositions that orient behavior. In this paper, we define a debugging competency as a constellation of mindsets, including the testing mindset, describe its relevance to engineering work, present approaches used in computer engineering courses to support student development of this mindset, and investigate evidence of the debugging mindset among students in these courses.

We define a debugging mindset based on several data sources: engineering and education literature, interactions with a departmental industrial advisory council, our own experiences and observations as engineering educators, and student written reflections. A student with a debugging mindset will commit to developing and executing a test plan to demonstrate that a design functions as envisioned. Once a test plan identifies an error in the system design, the debugging mindset involves a systematic approach to critically assessing system behavior, enumerating and testing hypotheses, and producing an updated understanding of the system. The debugging mindset exhibits an openness to various perspectives, navigating levels of abstraction, making connections, handling uncertainty, and updating one's mental model with new information.

In order to support such a debugging mindset among students taking computer engineering courses, we have developed activities that have been incorporated in a sequence of courses in the curriculum. We have taken this approach in response to (1) the varied ways mindsets can manifest in student work and (2) the necessity of repeated and diverse activities to facilitate change in some entrenched mindsets. A sophomore-level course with a lab project introduced system sketching, questioning and reflection activities before, during and after lab work and selectively in other course work. Lab demonstrations were expanded to including debugging as well as correct behavior. A junior-level course with a term project added well-defined testing and design roles that rotate among students in a lab group. Lab work includes explicit planning via an iterative, reflective team process.

This paper defines the concept of a debugging mindset in computer engineering. It presents approaches and observations from several semesters worth of two courses which targeted development of these mindsets among students. The paper concludes with an investigation of evidence of the debugging mindset among students in the two courses.

II. Background

Our work on debugging mindsets builds on literature both in mindsets and in debugging education.

A. Relevance of Mindsets in Education

A mindset can be described as a belief that orients thought and action [1]–[3]. For example, as Dweck describes, two of the most prominent mindsets of concern to educators are the growth and fixed mindsets. Both are described by the belief one has about intelligence and development. In a fixed mindset, one believes that intelligence and other basic qualities cannot be changed. In a growth mindset, one believes that "your basic qualities are things you can cultivate through your efforts, your strategies, and help from others" [1]. These

This material is based upon work supported by the National Science Foundation (NSF) under awards 1623125 and 2144757. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

beliefs affect a host of thought and action, such as how one responds to failure and effort one applies to challenging tasks.

While investigation of growth and fixed mindsets has been predominant in engineering education and related fields, others have studied more direct, profession-oriented mindsets. For example, several researchers have begun to identify and define a grouping of mindsets associated with design thinking. Here, mindsets are thought of as beliefs or stances that affect the way one approaches design work [2], [3]. In this space, mindsets are typically identified in sets of multiple mindsets that affect different aspects of one's approach. For example, Schweitzer et al. [4] explored eleven distinct mindsets, ranging from empathetic towards people's needs and context to inquisitive and open to new perspectives and learning to consciously creative. Dosi et al. [5] identified 22 total design thinking mindset constructs in their construction of a mindset self-awareness questionnaire.

Investigating mindsets in a more practical professional and task-oriented space helps differentiate mindsets from other constructs of interest such as skills and tools. While mindsets may affect skills—e.g., a fixed mindset may limit one's effort in learning a challenging new skill—they are distinct. Howard et al. [2] argue, in the context of design thinking, that focusing on skills and tools without mindset can be limiting. They argue that considering design thinking as a series of tools to enact limits consideration to "design doing". Similarly, in only considering skills, one loses appreciation for the nuance and context within which those skills are applied and what makes them effective toward a team's or organization's goals. Similarly, we argue that while debugging skills and tools are relevant to educators and students, considering debugging mindset(s) provides a more robust and contextualizable landscape of engineering student development.

Since mindsets focus on core beliefs, they can be challenging to change.

B. Debugging Education

Debugging has been long studied in education, particularly in the context of programming. McCauley et al. [6] reviewed over 50 papers that primarily focus on the sources and type of bugs in programs and development of skills for the debugging process. Specifically, the authors categorize educational debugging supports into reducing misconceptions of the system, improving comprehension of the programming language, and developing a systematic process for debugging. While most works focus on development of debugging skills, several works stand out in relation to the development of our debugging mindset and supporting activities.

Chmiel and Loui [7] aim to enhance students debugging skills by having students complete debugging exercises prior to developing programs for each assignment. Students were also required to track bugs they encountered in debug logs. The logs, in particular, were observed to aid students in both improving their approach to assignments, but also in their awareness of their growth.

Nagvajara and Taskin [8] state that there is a relation between debugging and non-technical competencies such as creative thinking, creative problem solving, and creativity in design. More precisely, these non-technical competencies can be considered prerequisites for debugging—i.e. a student must develop them to some degree before the student can acquire debugging skill. Bottcher et al. [9] also recognize that debugging is a complex skill and requires several non-technical base competencies to access it. In particular they identify that a software engineer needs awareness that bug detection can be done in a systematic way, needs strategies for finding and fixing bugs, and must ensure that the bugs can be detected in the future.

Lowe [10] explored debugging itself as an integrated tool to learn programming logic, language, and design. This work recognizes that debugging involves a mental model of the whole system at different levels of abstraction – from test cases and problem statement to application design to the source code to the notional machine. Therefore, it argues that taking a "debugging-first" approach can aide struggling novice programmers.

We are not the first to recognize the link between debugging and mindsets. In fact, McCauley et al. [6] identify Dweck's work as a potential avenue for exploring with respect to debugging. O'Dell [11] links the debugging mindset to the growth mindset and proposes a systematic process for debugging. Morales-Navarro et al. [12] leverage debugging to promote a growth mindset using the "debugging by design" technique where students design buggy projects for peers to solve. Despite the rich potential for a debugging mindset, there has been little work on actually defining and supporting a debugging mindset.

III. Debugging Mindset Definition

We argue for a mindset that is implicit in the skills seen as related to debugging and resonant with a growth mindset. We build such a mindset based on the above education literature on effective debugging, with consideration of two aspects: (1) observable behaviors that reflect effective debugging practice and (2) internal beliefs or worldviews that guide these behaviors.

People with more developed or expert debugging mindset approach unexpected system behavior or "bugs" with a systematic, iterative process of critical assessment, making and testing hypotheses, and drawing insights to better understand the system. While the systematic process starts with a holistic view of the system based on their current mental model, the debugging mindset is open to asking questions from various viewpoints and levels of abstraction. As these questions are answered through documentation, discussion, and experimental observations, the debugger's mental model is updated with new information and connections. This process is tenaciously followed until their mental model matches the observed behavior of the system. Those with a debugging mindset often explicitly plan for debugging both in the process and the design itself.

TABLE I: An initial debugging mindset rubric.

Mindset Element	Underlying Belief	Underlying Opposing Belief
Growth mindset	Failure is part of the learning process	If I have a bug, I did it wrong and I'm a failure
Testing mindset	If it wasn't tested, it doesn't work (or there are bugs)	If I wrote the code correctly, I don't need to test/if it passes basic testing, it's fine
Holistic/ system-level view	I see an entire system from different perspectives, at different levels of abstraction, and its various types of components and connections	I need to focus on what's in front of me (e.g., particular perspective, level, or component in a system)
Systematic process	Debugging is a process and I need to take a systematic approach	I prefer a trial-and-error approach
Inquisitiveness	It's fun to explore new bugs, I'm curious about the cause	Argh, bugs are a pain to find and address
Commitment to rigor	I need to find and address all the bugs	It just needs to be good enough; you'll never address all the bugs
Intrinsic value of debugging	Debugging is an essential part of the development process	Debugging is a waste of time and resources

Undergirding these observable behaviors is a core set of internal beliefs about debugging. Those with a well-developed debugging mindset believe debugging is an essential, guaranteed, and positive part of the system design process. These debuggers accept that their mental model is useful, yet imperfect—bugs will exist. Finding these bugs is an exciting process of growth and learning in which to actively engage. This excitement is aided by a confidence that the buggy behavior will be explained since all buggy behavior has a root-cause and is not "just happening". The debugging mindset does not view the absence of the buggy behavior as the sole outcome of the debugging process, but views the updated mental model and richer cache of bugs as critical outcomes.

We also identify the many observable behaviors that show a novice or undeveloped debugging mindset. These are important to recognize in order to target support for growing a debugging mindset in students.

Those without a well-developed debugging mindset tend to immediately jump into their debugging process with a focused trial and error approach. They often repeat the exact same experiments multiple times, yielding the same (confounding) results. When examining the results, they may not actually consider other perspectives or levels of the system/process such as whether the test and expected results are correct. They quickly express frustration at the slow or non-existent progress—i.e., no bugs are fixed so there is no progress. This frustration may lead to turning to external help for the complete solution, including asking instructors for the "answer" (i.e., bug localization and fixing) rather than help updating their mental model. More drastic still, students may give up or turn to copying others' solutions.

This "anti-debugging mindset" has corresponding internal beliefs that are critical to consider when trying to inculcate students with a debugging mindset since these may represent fundamental roadblocks to developing a debugging mindset.

Someone with an under-developed debugging mindset views bugs as a failure and thus bad. Debugging causes them extra, perhaps unplanned, work and is, thus, a punishment for their failure. Often they lack confidence in fixing the issues and have corresponding uncertainty and fear around how long a bug will take to find and correct. Such novice debuggers are prone to externalizing the blame for the bug—a tool issue, co-worker/partner issue, documentation issue, or lab manual issue.

Those with a under-developed debugging mindset also view the debugging process less iteratively—once a bug is fixed, they expect the whole system to work rather than appreciating that the system may have more bugs or their fix may have introduced bugs. These students believe debugging is about getting the "right answer" to the specific issue (i.e., fixing the bug) rather than considering the system and process holistically.

As an aid for potential evaluation of debugging supports we have proposed an initial debugging mindset rubric (see Table I).

IV. Case Studies in Supporting a Debugging Mindset

Debugging as a critical mindset and the resulting prototype activities to support it emerged from instructional teams, referred to as x-teams, for two courses — an embedded systems course and a computer organization course. The x-team model is being developed and used as part of an NSF Revolutionizing Engineering Departments (RED) grant [13]–[16]. An x-team includes and supports the instructor of a course. It leverages practices and tools from design thinking and engages in collaborative design and reflection to implement student centered teaching approaches [16]. Design thinking tools that have helped x-teams better address student needs include empathy maps, personas, and journey maps. In this section we describe the course-specific context, motivation, and prototype activities to support debugging mindsets in these two courses.

A. Course 1: Embedded Systems

The Introduction to Embedded Systems course is a 200-level course in computer, cybersecurity, electrical and software engineering majors. The course introduces students to hardware and software aspects of embedded systems including microcontrollers, memory, input/output interfaces, embedded programming in C, initialization and configuration of peripherals, polling and interrupt processing, and mobile robots. The first third of the course covers foundational concepts and skills; the middle third, understanding and using microcontroller peripherals; and the final third, implementing a project in the lab for an autonomous vehicle application. Among the learning objectives for the course are designing and debugging applications on embedded platforms, gaining familiarity with the field of embedded systems, and considering socio-technical factors in engineering work and solutions.

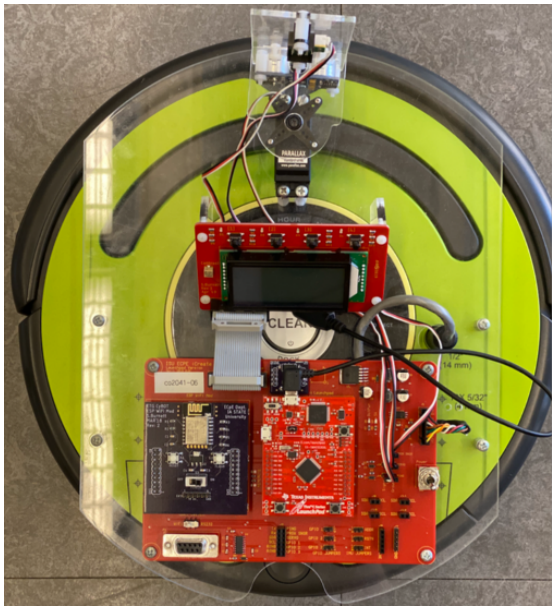

Fig. 1: Custom mobile robot platform.

The mobile robot in the lab is an iRobot Create 2 (Roomba compatible) and can be controlled with Open Interface commands from a microcontroller board, as shown in Fig. 1. The microcontroller board interfaces to input/output devices added to the robot, including an infrared sensor, ultrasonic sensor, and servo motor (used to get scans of sensor readings). In the final project, teams program the microcontroller to move the robot through a test field and avoid obstacles to reach a destination.

Students, having wide-ranging backgrounds and skills, introduce many different types of bugs, especially given the complexity of the embedded system and lab work. Debugging has not been extensively taught in the course or curriculum, however, guidance is provided to teaching assistants and students in the course. General debugging steps and tips, based on hypothesis-based approaches, are outlined for students in the first lab. Teaching assistants are trained to encourage students to think about and talk through a problem and take key steps. Also in the first several labs, debugging exercises that introduce debugging tools and techniques in the development environment are included in the lab manual.

Instructors and teaching assistants in the course observed that many students did not take a systematic approach to debugging, lacked confidence in debugging skills and tools, and found it difficult to ask useful questions as part of understanding a problem and searching for errors. Most of the observable behaviors of the debugging mindset were not evident, or at least easily seen, in interactions with students in the lab. Thus we designed new course activities to both elicit and observe behaviors. The activities were designed to promote the development of problem solving and learning strategies associated with debugging. As O'Dell concludes: "Since solving bugs requires learning, the debugging process can be made easier by better understanding effective learning and teaching strategies." Like O'Dell, we also wanted students to change the way they perceived the challenge of debugging. We placed an emphasis on mental models, curiosity and questioning and reinforced this in various ways throughout the course. In addition, as a part of every lab demonstration, students not only demonstrated functionality but also debugging. This was intended to foster the beliefs that debugging is part of the learning experience and is essential to the development process.

New activities were initiated during spring semester 2020, but the pandemic curtailed their full realization once students could no longer go into the lab to experiment. However, their initial use showed some promise, and they have been used again in subsequent semesters. The activities were designed by an instructional team and informed in part through research about questioning [17], [18]. This resulted in five new activities:

1) Debugging and Q&A demonstrations added to the traditional lab demonstration
   - Completed weekly with every lab using evaluation rubrics
2) Lab planning in-class activity based on the Question Formulation Technique (QFT)
   - Facilitated early in the semester with a worksheet and then used informally
3) Postlab documentation of questioning and debugging work
   - Completed every two weeks as part of a postlab worksheet
4) Writing your own homework question on a concept of your choice
   - Submitted with homework assignments (about every three weeks)
5) Applying the five whys method to improve problem solving
   - Suggested practice with homeworks and quizzes and offered as extra credit for exams and the lab project

Every week, like traditional labs, students demonstrated lab functionality to a teaching assistant. In addition, they also completed a debug demo and a Q&A demo, worth the same number of points as the functional demo. For the debug demo, they were instructed to use debugging to explain something about the internal workings of their system. It was evaluated based on these criteria: (1) using basic debugging functionality and views in Code Composer Studio; (2) giving a specific example of program execution and/or a problem in the code; (3) providing a specific example of testing a change; and (4) explaining system operation using specific information in one or more views. For the Q&A demo, they were expected to (1) formulate a useful question, such as a priority question during lab planning or a question created while doing the lab; and (2) generate relevant and appropriately focused questions that have some purpose, such as to clarify a topic, understand a relationship, make connections, explore what is and is not

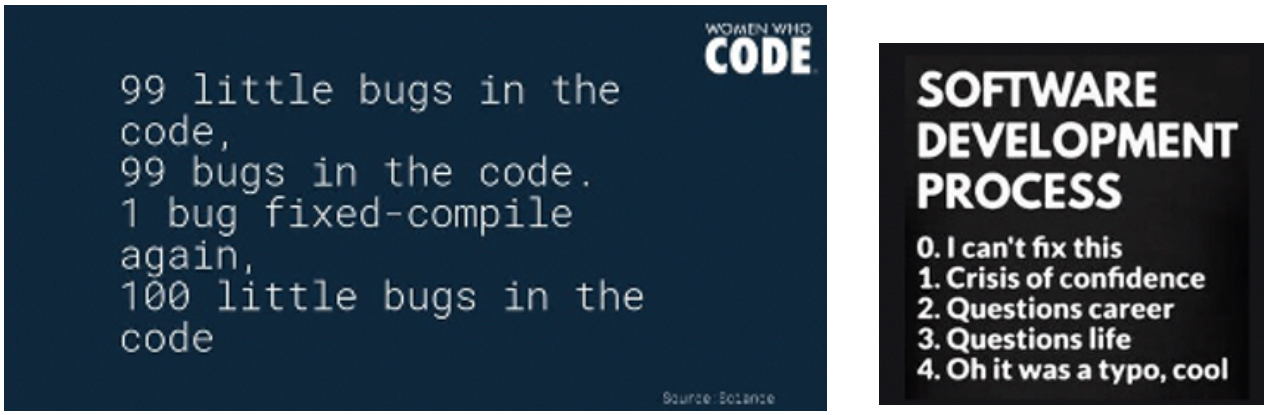

Fig. 2: Sources: https://twitter.com/womenwhocode (left) and unknown (right).

known, make judgments, express curiosity, challenge ways of thinking, test new ideas, etc.

Postlab assignments were less frequent, but required students to document and reflect on lab work. It consists of three parts: (1) prelab planning notes, such as questions from the QFT lab planning activity; (2) lab notes, such as follow-up on their questions, description of debugging, and any updates to their system sketches, i.e., mental models; and (3) a reflection on their lab experience. Once again, questioning and debugging are incorporated as key elements of the learning experience.

In addition to introducing the Question Formulation Technique, we incentivized using the five whys method for structured questioning and root cause analysis. As an industry practice, it resonates with students, and they can quickly apply it and see some benefits. In the course, students were rewarded not only for a correct answer but also for their work to go from an incorrect to a correct answer.

Beyond the assignments and activities in the course, the instructor and teaching assistants tried to promote and model the mindset behaviors and attitudes in class, lab and through online platforms (e.g., Canvas and Piazza). Questions, bugs, and debugging stories were shared through Piazza. Motivational quotes and cartoons were interspersed on Canvas pages, such as the examples shown in Fig. 2.

B. Course 2: Computer Organization

The Computer Organization and Assembly Level Programming is a 300-level that follows Course 1 in the curriculum. Student feedback collected through two offerings suggested that the significant lab experience in the course required a very large time investment that many students felt was overwhelming. In particular, they struggled with becoming familiar with the hardware description language (HDL) and simulator used in the course. Even eager, engaged, and self-effective students described parts of the lab experience as "drinking from a firehose." Further, students felt that the testing requirements were uselessly high and tedious. Finally, there was a frequent notion of feeling lost trying to debug problems (e.g., identifying whether the issue was a conceptual issue in the design or testing, an HDL description issue, or a simulation tool issue). This was reflected in a large fraction of self-reported time being devoted to "debugging".

We iteratively developed a series of activities aimed at improving students' experience with the challenging term lab project while inculcating a debugging mindset. Of course, no one-shot activity was going to boost the various aspects of a debugging mindset alone. Therefore, over the course of four offerings of the course we added a range of activities, big and small, that repeatedly reinforced a debugging mindset. These activities targeted two aspects – motivational and skills-building. Motivational activities aimed to increase students' perceived value of testing and debugging both in the short- and long-term. Skills-building activities aimed to build both domain-specific testing and debugging skills as well as the systematic habit of testing and debugging as an intrinsic part of the design cycle.

In past offerings of the course students were required to develop their own test plans and justify that their system designs worked in a report. This led to a common perception, handed down between course offerings, that "no one's processor works." Long, tedious hours in the lab were spent on useless work, ultimately destined for failure. Whether or not this was ultimately true, it made students entering the lab unmotivated to engage with the lab experience. Furthermore, if this was really how the students leaving the course felt, it suggests that even if they had learned a lot during the course they would not be willing to access that knowledge going forward. In order to address this we now spend time during lecture sessions devoted to understanding the role of testing in computer hardware design industry (e.g., test and debug engineers) and looking up the silicon errata of the "not working" processors students are actively using in their personal computing devices. We pair this with a open-sourced (to students) testing framework that models a systematic testing approach. Students collaboratively as a class built up test sets for their processors. The goal is to motivate students to be able

to demonstrate that their processor works to some level and to be able to articulate where their processor's functionality is wrong or unknown.

The introductory labs assignments were redesigned to provide students with direct experience with a design cycle specific to the course. This design cycle explicitly included testing and debug as first-class components, critical for success in the course. Initially, students were walked through this design cycle using a moderately complex design. Students must fill out small portions during the testing (e.g., test case inputs/outputs and complete testbench code) and debug steps (e.g., fix the HDL of the design). In particular, the provided design files contain three distinct bugs representative of common errors likely to be seen throughout the course. This not only practically models where the debugging fits into a design cycle but aims to help students internalize that debugging itself is a mechanism of learning more about a system – in this case a relatively unknown system.

While domain-specific debug tips were sprinkled throughout the first few labs at timely locations in an attempt to ease pain-points, no formal debugging module was taught since the goal was to repeatedly reinforce the debugging mindset. Therefore, the design cycle was continued throughout the duration of the course lab experience in required team contract planning, lab report entries, submitted testbenches, and individual feedback forms for each lab assignment.

Team contracts reinforced the debugging mindset in two ways. First, they require students to explicitly assign a separate person to test each subcomponent. To students this is motivated as a mechanism to reduce debug time by increasing the number of perspectives of the system being tested (and thus what may be wrong with it during debug) as well as by ensuring that submodules are tested before integration. Second, the team contract/lab manual examples demonstrate the addition of time for debugging the system, reinforcing the notion that debugging is an expected part of the design cycle and learning rather than a negative result of failed understanding.

Our lab report templates require students to demonstrate that each of their subcomponents were working with a simulation waveform and description. Students also had to justify that their tests were reasonably comprehensive. Additionally, the lab report waveforms must be accompanied by testbench files for each subcomponent, emphasizing the systematic approach to testing and allowing for future use during system-level debugging.

After each lab assignment was turned in, students filled out an individual feedback form which included a log of time spent on various tasks, including debugging. It also asked students to identify the largest challenges they faced and how they could overcome such challenges in the future. The hope was that this would help increase students' awareness of what caused their bugs (i.e., their misunderstanding of the system or process) and how to fix or avoid them going forward (i.e., update their mental model of the system or process).

Most recently we have included a course-specific "bug report" (shown in Fig. 3) in early labs that explicitly leads

Quick description (< 10 words) Include answers to the following:

1) What was the system setup?
2) What are the inputs (files, signal values, etc.)?
3) What were the expected results (print outputs, signal/waveform behavior, etc.)?
4) What were the actual observed results (print outputs, signal/waveform behavior, etc.)?
   - Include screen captures of waveforms
   - Include log transcript as a file
   - Include code or minimum working example?
5) What, if any, errors exist?
6) What, if any, warnings exist? Why are they ok to ignore?
7) Any other behavior you see that you think may be relevant
8) Attempts made to solve/debug this problem – you should have tried at least one, and include an explanation of your thought process

Fig. 3: Bug report template.

students through a systematic thought process. These are used to solicit aid from fellow labmates, TAs, and course instructors. Each student must submit at least one bug report throughout the course of the introductory labs and substantively respond to another student's bug report. Finally, they must post how their bug was resolved. The goal with this approach is to provide students with a systematic approach defining the problem/issue to debug and to explicitly identify differences between their internal model of the system and the observable output from the actual system. During the bug report students are required to have attempted at least one experiment to resolve the issue and analyze what happened in that case. Given the systematic nature of the bug report and fact that it offered a mechanism to get assistance from others, we believe that it would encourage students to persist in fixing bugs and gain alternative perspectives on what could be the cause of bugs.

V. Analysis

A. Methods

We utilized directed content analysis [19] to investigate connections to the debugging mindset evidenced by students in each of the courses. Directed content analysis acknowledges that existing theory may be emerging or incomplete, as in the case of understanding a debugging mindset among engineering students, and seeks to better identify or extend the nascent theory. Here, we leverage the conceptualization of a debugging mindset described earlier and explore elements, expansions, and modifications of that conceptualization evidenced in student reflections on lab and project work.

We employed the design research tool of empathy maps [20] to support content analysis. An empathy map organizes observations of an individual or group based on four key facets of an experience: (1) what they do or say, (2) what they see or hear, (3) what they think, and (4) what/how they feel. Together, these aspects provide a brief, multi-faceted overview of an individual or group's experience. In this case, the empathy maps reflect the experience of debugging within the courses based on several data sources: student reflections, instructor observations, and other course artifacts.

Three researchers conducted analysis for Course 1 and two different researchers conducted analysis for Course 2. Researchers iteratively moved through the following stages:

- Compile and familiarize self with data
- Identify data excerpts with potential connections to the debugging mindset
- Code excerpts for evidence of debugging mindset or opposing mindset
- Individually, place coded excerpts within an empathy map
- Collaboratively within each case, merge individual empathy maps to reflect key themes in students' experience of debugging mindsets
- Describe key themes in a narrative and create final empathy maps

B. Course 1: Embedded Systems Empathy Map

Table II shows a summarized composite empathy map for the embedded systems course generated from the team's review and analysis of student work. The student work used in the analysis includes several assignments from spring semester 2022: postlab assignments, midterm survey, and final lab project survey. Debugging was explicitly addressed in the postlab assignments via the debugging demonstration and commonly reflected on in other sections. Student comments in the surveys also directly and indirectly referred to debugging given its emphasis in the course. Using these assignments, we found all elements of the debugging mindset shown in Table I. There were many examples that support the beliefs behind each element, and a few examples of opposing beliefs as well.

The growth mindset appeared in comments about time, iterations and engagement needed for debugging, as well as the value of partners or teammates in the process. Students also held the view that more effort on their part in advance (e.g., more closely reading the lab manual, refreshing their programming skills) would eliminate some bugs. While some students were frustrated with the amount of time or information needed, a sense of failure due to bugs was not evident other than a perception about falling behind compared to others (though they eventually realized others weren't farther ahead). Students realized that many components in the system can affect its operation and that focusing too quickly or narrowly may not solve a problem, thus exhibiting a system-level view. They sometimes added new information to their system sketches and connected concepts to practice. Both inquisitiveness and rigor are fostered with better mental and conceptual models of the system. Through the project, students generally embraced debugging as valuable to being an engineer. As students gained knowledge and experience during the course, they took more systematic approaches to debugging, including how they used the debug tool. Nonetheless student work also shows that some students take a trial-and-error approach, fix just enough bugs, and test only basic functionality, limiting the debugging mindset.

TABLE II: Summary empathy map for Course 1.

Do/Say	**See/Hear**
• Described some features of the debug tool that were used • Referred to debugging tasks, such as trial and error, printing, stepping through, watching variables, commenting out code, etc. • Cited importance of lab documentation and resources, such as manuals, prelabs, class notes, and need to use more effectively before lab to reduce time spent completing the lab (i.e., while debugging, found helpful information that was provided) • Used an iterative process over time (testing and retesting, writing and rewriting code) • Developed solutions to unforeseen issues • Asked TAs questions and interacted with TAs in the debugging process • Students start with what's familiar, or what they think they know	• Debugging provided a way to identify code, system, and mental errors and try to fix them • Debugging was time-consuming and even simple issues took a lot of time to fix • Time taken to understand the components was well spent • Disciplined approach led to successful debugging outcomes • Being too narrowly focused led to overlooking key information in other resources • Debugging was helped by more information and perspectives • Communication with and roles of partners/teammates affect debugging performance
Think	**Feel**
• Better problem solving skills • Important to understand equipment and the overall system, not just the code • Embedded programming is both more interesting and more difficult and takes more time to get right • Debugging brings insights and/or identifies gaps that help "connect the dots" in the course and learn more about embedded systems • There is value in working with others • Useful to have a continuous debugging approach, i.e., start small with working code • Reading more lab information beforehand will reduce debugging time • Code should be thoroughly understood before using it • Better hardware will make a system perform exactly as written in code	• Liked using the debug tool to check code status, e.g., variables, conditions, control flow, registers, etc. • Enjoyed the challenge of debugging • Felt like an engineer when solving problems including debugging • Frustrated sometimes, e.g., time spent, uncertainty about the issues, incomplete information, initial struggles. • Confused sometimes, e.g., complex system, physical as well as software issues, various technical and educational materials, unfamiliarity or inexperience with lab • Felt rushed, e.g., finding errors in the testing area, finishing a lab before the next lab, falling behind compared to others • Satisfaction with finding, fixing and understanding bugs

C. Course 2: Computer Organization Empathy Map

Table III shows a summarized composite empathy map generated from the team's reading of an intermediate individual feedback form from the term project, a open-ended individual evaluation of bug reports, and the final team report from the term project. Overall, debugging and testing were very common topics of discussion even for questions/assignments that did not specifically mention debugging/testing. This is not surprising as students also indicated that a significant

TABLE III: Summary empathy map for Course 2.

Do/Say	See/Hear
• Overcame (debugging) challenges to complete tasks • Recognized root causes of failing debugging tasks or taking too much time on debugging tasks, especially after large time-lost error • Frequently sought external help (perhaps valuing perspectives) • Frequently put off systematic debugging tasks to end • Spent much of time debugging without robust mental engagement • Debugging forced (independent) learning	• Debugging was contextual – needed to learn tools/methods specific to task • Actual debugging time longer than expected • System-level integration debugging challenging/time-consuming • Teammates significantly affected debugging success/time • Testbenches and debug reports (two parts of debug infrastructure) points of reflection/consideration
Think	**Feel**
• I need stronger background knowledge to support debugging • Debugging tasks can be simplified by design / implementation choices • Debugging and testing is a time-consuming, but inherent part of the process • If the whole system is working, other tests or warnings are irrelevant • Systematic and holistic debugging approaches can help debugging success • Many tasks associated with a systematic testing/debugging approach are wasted work (either for project or learning)	• Excited at the experience of debugging, especially working through strange/unique errors • Frustrated at time and effort of debugging • Debugging is pain • Anxious at errors, especially in a time crunch • In control once errors are understood and corrected • Confused, helpless, or floundering during the process of debugging

amount of student time was expended on debugging and testing aspects. Importantly, using the above assignments we were able to observe all aspects of the debugging mindset rubric shown in Table I to some extent.

Many debugging-related student comments touched on the growth mindset aspect. Students thought they needed more background knowledge in the specific course context to support their debugging (e.g., knowledge of VHDL, assembly programming, or digital logic simulation tools and associated errors). Despite this perception, students across the spectrum of project completion overcame debugging challenges to complete significant portions of the project. Many stated or demonstrated that debugging forced them to independently learn new material and concretely connect it to course content. Students often root-caused their struggles with debugging. The most common issues were underestimating debugging time and not using systematic or holistic debug approaches. Although these students demonstrated a growth mindset and some reported learning that they were capable of such independent learning, many comments did not reflect an internalization of the intrinsic role debugging had on their learning.

Students frequently commented on testing in relation to its role in debugging. Both of these tasks were reported as time-consuming with frustration and anxiety over perceptions of wasted work. Despite this frustration, students often acknowledged that debugging and testing were an intrinsic or "vital" part of the design process. Evidence of this acknowledgement is a change from semesters prior to the added activities.

Interestingly, while students frequently started out with some level of confusion and anxiousness and endured a time of pain or discomfort, once the bugs were found and eliminated they reported a sense of control and excitement. Indeed, many student comments reflect a knowledge of how many bugs remained (i.e., testing mindset) and a desire to see them eliminated if more time were available (i.e., commitment to rigor).

VI. Conclusion

Empathy-map-aided content analysis of student reflections provided evidence of the proposed debugging mindset in each of the course cases. This evidence spanned the seven constituent mindsets described in Table 1, in both affirmative and opposing beliefs. New constituent mindsets were not identified but novel conceptualizations and manifestations of the extant constituent mindsets were evident in the final empathy maps. This evidence highlights the relevance of the proposed debugging mindset to engineering student work, particularly in the given contexts of computer engineering laboratory and project work, and the opportunity for future work to (1) explore the relevance of the debugging mindset in other engineering student contexts, (2) better understand what a debugging mindset and its constituent components look like in an engineering student context, and (3) develop qualitative and quantitative instrumentation to support educators and researchers in identifying the debugging mindset among engineering students.

While the analysis was not designed to identify causal links between course pedagogy and the development of debugging mindsets among students in each course case, there was evidence of whether and how students' debugging mindsets could change within a course. For example, the Course 2 empathy map demonstrated that students often recognized causes for "debugging failures" when they spent substantive amounts of time working through errors. Latent clues to the growth this points to is evident in complementary empathy map items that demonstrate negative experiences in the process of debugging and positive experiences based on outcomes of debugging. Strong conclusions should not be drawn from this evidence, but it does point to further work that should be completed to better understand how students develop debugging mindsets and how educators can support such development.

References

[1] C. S. Dweck, *Mindset: The new psychology of success*. Ballantine Books: New York, 2016.

[2] Z. Howard, M. Senova, and G. Melles, "Exploring the role of mindset in design thinking: Implications for capability development and practice," *Journal of Design, Business & Society*, vol. 1, no. 2, pp. 183–202, 2015.

[3] H. G. Nelson and E. Stolterman, *The design way: Intentional change in an unpredictable world.* MIT press, 2014.

[4] J. Schweitzer, L. Groeger, and L. Sobel, "The design thinking mindset: An assessment of what we know and what we see in practice," *Journal of Design, Business & Society*, vol. 2, no. 1, pp. 71–94, 2016.

[5] C. Dosi, F. Rosati, M. Vignoli *et al.*, "Measuring design thinking mindset," in *DS 92: Proceedings of the DESIGN 2018 15th International Design Conference*, 2018, pp. 1991–2002.

[6] R. McCauley, S. Fitzgerald, G. Lewandowski, L. Murphy, B. Simon, L. Thomas, and C. Zander, "Debugging: a review of the literature from an educational perspective," *Computer Science Education*, vol. 18, no. 2, pp. 67–92, 2008.

[7] R. Chmiel and M. Loui, "An integrated approach to instruction in debugging computer programs," in *33rd Annual Frontiers in Education, 2003. FIE 2003.*, vol. 3, 2003, pp. S4C–1.

[8] P. Nagvajara and B. Taskin, "Design-for-debug: A vital aspect in education," in *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)*, 2007, pp. 65–66.

[9] A. Böttcher, V. Thurner, K. Schlierkamp, and D. Zehetmeier, "Debugging students' debugging process," in *2016 IEEE Frontiers in Education Conference (FIE)*, 2016, pp. 1–7.

[10] T. Lowe, "Debugging: The key to unlocking the mind of a novice programmer?" in *2019 IEEE Frontiers in Education Conference (FIE)*, 2019, pp. 1–9.

[11] D. H. O'Dell, "The debugging mindset: Understanding the psychology of learning strategies leads to effective problem-solving skills." *Queue*, vol. 15, no. 1, pp. 71–90, 2017.

[12] L. Morales-Navarro, D. A. Fields, and Y. B. Kafai, "Growing mindsets: Debugging by design to promote students' growth mindset practices in computer science class." in *Proceedings of the 15th International Conference of the Learning Sciences-ICLS 2021*, 2021.

[13] N. D. Fila, S. McKIlligan, and K. Guerin, "Design thinking in engineering course design," in *2018 ASEE Annual Conference & Exposition*, 2018.

[14] N. D. Fila, S. McKIlligan, and S. Abramsky, "How engineering educators use heuristics when redesigning an undergraduate embedded systems course," in *2018 ASEE Annual Conference*, 2018.

[15] N. D. Fila, D. Rover, M. Mina, and P. Jones, "Designing a course together: A collaborative autoethnographic study of a cross-functional team course design project in engineering," in *2020 ASEE Virtual Annual Conference*, 2020.

[16] S. McKilligan, N. Fila, D. Rover, and M. Mina, "Design thinking as a catalyst for changing teaching and learning practices in engineering," in *2017 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2017, pp. 1–5.

[17] "Qft teaching and learning resources," Right Question Institute. [Online]. Available: https://rightquestion.org/education/resources/

[18] S. Moaveni and K. Chou, "Using the five whys methods in the classroom: How to turn students into problem solvers," *Journal of STEM education*, vol. 17, no. 4, 2017.

[19] H.-F. Hsieh and S. E. Shannon, "Three approaches to qualitative content analysis," *Qualitative health research*, vol. 15, no. 9, pp. 1277–1288, 2005.

[20] D. Gray, "Updated Empathy Map Canvas," Jul. 2018. [Online]. Available: https://medium.com/the-xplane-collection/updated-empathy-map-canvas-46df22df3c8a