Developing a Real World Escape Room for Assessing Preexisting Debugging Experience of K12 Students

1st Tilman Michaeli
Computing Education Research Group
Freie Universität Berlin
Berlin, Germany
tilman.michaeli@fu-berlin.de

2nd Ralf Romeike
Computing Education Research Group
Freie Universität Berlin
Berlin, Germany
ralf.romeike@fu-berlin.de

***Abstract*—Debugging code is a central skill in learning to program. Nevertheless, debugging poses a major hurdle in the K12 classroom, as students are often rather helpless and rely on the teacher hurrying from one student-PC to the other. Despite this, debugging is an underrepresented topic in the classroom as well as in computer science education research, as only few studies, materials and concepts discuss the explicit teaching of debugging. According to the constructivist learning theory, teaching and developing concepts and materials for the classroom have to take learners' preexisting experience into account. Students' preexisting debugging experience is built through troubleshooting, where they frequently find and fix errors in their daily lives – before they learn to program – for example when repairing their bicycle or if "the internet" stops working. Debugging is a special case of general troubleshooting and shares common characteristics, such as the overall process or particular strategies. Thus, the aim of this study is to develop an instrument for assessing preexisting debugging experience in the form of a real-world escape room consisting of debugging-related troubleshooting tasks. This allows us to observe students' troubleshooting process, strategies, and overall behavior in a natural environment and thus assess preexisting debugging experience. To this end, a design-based research process was conducted and a real-world escape room consisting of various troubleshooting tasks was developed. Those tasks and the escape room setting provide an innovative methodological approach to study students troubleshooting behavior and assess their preexisting debugging experience.**

***Index Terms*—debugging, troubleshooting, escape room, computer science education, K12, computational thinking**

I. Introduction

In programming, systematically examining programs for bugs, finding and fixing them is a core competence of professional developers who spend between 20% and 40% of their working time on debugging code [1]. In the K12 classroom, however, debugging poses a key problem: For learners, fixing programming errors is not only a significant obstacle to learning programming [2], but is additionally a major source of frustration [3]. Teachers, on the other hand, lack adequate concepts and materials for fostering and addressing debugging in the classroom. Most of the time they rush from one student to another and attempt to support them individually [4], an approach best described as "putting out the fires". In consequence, novices are often left alone with their errors and forced to learn debugging "the hard way" – similarly to how many professionals have learned debugging [1].

To eventually develop suitable concepts and materials to address this problem in the classroom, learners' preexisting experience has to be taken into account: According to the learning theory of constructivism, learning is a constant and active process of refining preexisting models of a subject by making and reflecting on new experiences [5]. The importance of assessing such preexisting experiences is emphasized in the concept of educational reconstruction [6] as well. While such preexisting experience for various topics is researched in computer science education frequently, this does not apply to debugging. Therefore, the aim of this study is to develop an instrument to assess preexisting debugging experience of K12-students.

But what is preexisting experience on debugging? In their daily lives, students are confronted with errors long before they build programming experience: Whether there is a problem with "the internet" or with their bicycle, they are troubleshooting and locating and fixing errors. Debugging is a special case of general troubleshooting and shares common characteristics, such as the overall process or particular strategies [7]. This is also reflected in the concept of computational thinking, for which debugging is one such approach [8] and hence considered important for every student. Therefore, in line with Simon et al. [9], we consider real-world troubleshooting experience as preexisting debugging experience that needs to be incorporated when teaching debugging. To this end, this paper describes the development of an instrument to assess such preexisting experiences by adapting an escape room approach.

The paper is structured as follows: Firstly, the theoretical background is analyzed, where the link between troubleshooting and debugging is established. Then, related work regarding assessing preexisting experience in computer science education in general, and debugging in particular, as well as the usage of escape rooms as a research methodology is discussed.

Building upon this, in section 4 we outline the research process of developing the escape room and the design criteria. Afterwards, we present the resulting escape room instrument and its tasks. In section 6, challenges, and the respective adaptions to the escape room, are discussed based on the evaluation. Finally, in section 7 we present our conclusions.

II. Theoretical Background: Problem Solving, Troubleshooting and Debugging

A. Problem Solving and Troubleshooting

Problem solving is defined by Anderson and Crawford [10] as "any goal-directed sequence of cognitive operations". Which skills are required to be a "good" problem solver depends strongly on the problem, which is characterized by various factors such as abstractness, structure, or success criteria. Jonassen [7] identifies eleven different classes of problems according to these criteria, such as algorithmic, logical, or decision problems. One of the problem classes is the class of troubleshooting problems:

> "Troubleshooting is among the most common forms of everyday problem solving and in many domains is synonymous with problem solving, perhaps because the inoperative entities that involve troubleshooting are most easily perceived as problems. Mechanics who troubleshoot your inoperative car or computer programmers who debug your inoperative computer are always recognized as problem solvers. The primary purpose of troubleshooting is fault state diagnosis. That is, some part or parts of a system are not functioning properly, resulting in a set of symptoms that have to be diagnosed and matched with the user's knowledge of various fault states."

Troubleshooting thus refers to the process of locating the cause of a system malfunction and then repairing or replacing the faulty component [11]. Debugging is troubleshooting in the domain of programming, a special case of general troubleshooting [12]. As further examples (and thus other domains) of troubleshooting problems, Jonassen [7] lists the identification of chemicals in a sample with a qualitative analysis, the identification of communication breakdowns in a committee, or the determination of why newspaper articles are poorly written. This reveals a holistic understanding of troubleshooting that goes beyond "technical troubleshooting" in the domain of engineering.

In the following, the general troubleshooting process and corresponding strategies are outlined. Building upon this, in the next section troubleshooting can be compared to debugging. This way, the strong correlation between debugging and troubleshooting will be introduced, and troubleshooting as pre-existing debugging experience will be theoretically grounded.

The general troubleshooting process can be summarized as follows (see [13] or [14] and figure 1). First, the actual problem has to be specified. To this end, a mental model of the system and its components must be created and both the incorrect and the correct behavior of the system has to be identified.

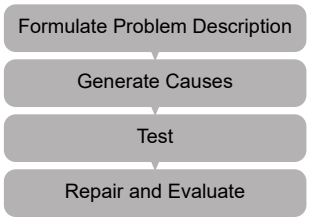

Fig. 1. Troubleshooting process according to Schaafstal et al. [14].

Based on this, hypotheses about possible causes are developed, which incorporate previous experience (*recognition-primed decision making* according to [15]). These hypotheses must then be tested, and the hypotheses either accepted, rejected, or refined iteratively. Certain strategies support this process of generating and verifying hypotheses. Finally, the error is corrected and tested again to verify whether the correction is actually successful and/or if there are other errors. For example, [16] confirms this process in a think-aloud study for the domain of circuit design.

The general troubleshooting process is supported by applying strategies. Jonassen and Hung [13] identify two levels of such strategies:

- *Global strategies:* Strategies that are independent of the concrete domain or system and help to limit the problem space. Examples include *Trial-and-Error*, *topographic search*, binary search, or *functional/discrepancy detection*.
- *Local strategies:* Strategies that are specific to a domain or system.

There is a large number of studies dealing with troubleshooting in particular domains, such as electronic circuits [17], [18], production systems [19], [20], or radar systems [14]. They identify local and global troubleshooting strategies relevant in the respective domains, and also show that global strategies vary in significance depending on the domain.

B. Debugging as a Special Case of Troubleshooting

As Debugging is a special case of troubleshooting in the domain of programming, similar skills are required [21]. This becomes evident when comparing the general troubleshooting and the debugging process: In both debugging and troubleshooting, the corresponding system (i.e. the program code) must first be understood, starting from the system's malfunctioning (debugging: the program). In the next step, information and clues must be collected to guide the generation of hypotheses. Those hypotheses then have to be tested, and finally, the error (debugging: bug) has to be corrected (see for example [12], [22]–[24]). Therefore, debugging follows the same iterative process of repeatedly formulating, verifying, and refining hypotheses until the cause of the error is found.

The same applies to strategies that support the general debugging process. Examples of local strategies for debugging are *slicing*, *print*-debugging or test cases. For many of the

local debugging strategies, there are corresponding global troubleshooting strategies. For example, for the local strategy of forcing the execution of a particular case and comparing the actual program output with the expected output, *functional/discrepancy detection* can be considered the corresponding general global troubleshooting strategy [21]. Similarly, the strategy of *print*-debugging for step-by-step tracing of a program flow can be considered the local counterpart of a global *forward topographic search*.

However, on the topic of transferring skills (such as computer science-related approaches and concepts according to Computational Thinking), results have often been underwhelming [25]. In contrast, indications for the transfer of debugging skills beyond the domain of programming have been shown to exist in a study by Klahr and Carver [26]. They gave students one hour of debugging training as part of a larger Logo curriculum. It contained a flowchart characterizing the debugging process, bug mappings, and debugging "diaries" that were always present in the classroom. Besides an improvement in students' code debugging skills, the authors found improved performance, such as higher accuracy and a more focused search in the non-computer transfer tasks.

In summary, we see that debugging is a special case of general troubleshooting. Both the systematic debugging process and debugging strategies like tracing or testing (*local strategies*) are manifestations of the general troubleshooting process or *global troubleshooting strategies* like *topographic search* or *functional/discrepancy detection*. Previous research results indicate that a transfer can actually take place. Given those similarities, troubleshooting can be seen as preexisting debugging experience, in particular with regards to the general process and certain strategies involved.

III. Related Work

A. Measuring Preexisting Skills in CS and Debugging

According to the learning theory of constructivism, learning is a constant and active process of adapting existing mental models by making and reflecting on new experiences [27]. Therefore, the already existing experiences of learners must be taken into account for the development of appropriate concepts and materials for teaching [5], [28]. Such experiences – often from real life and pre-teaching – generally include cognitive, affective, or motivational factors. For the design of specific teaching-learning settings, concrete domain-specific preexisting experience (such as preconceptions, prior knowledge, or ideas) is particularly relevant. In computer science education, there are numerous studies of such domain-specific preexisting experiences on different topics. In the field of programming, Onorato and Schvaneveldt [29] and Miller [30] determined learning preconceptions by observing learners during "programming" in natural language. Gibson and O'Kelly [31] analyzed the students' problem-solving process for search problems in a similar way in a classroom setting, while Kolikant [32] examined their preconceptions concerning parallelism and synchronization using written assessment. Similar to this, in the "commonsense computing" series, preexisting experience for various topics such as sorting or logic were investigated by using written assessments as well [33]–[35].

The study conducted by Simon et al. [9] within the commonsense computing series is central to this paper. They investigated preexisting debugging experiences for university students by analyzing their troubleshooting behavior to conclude implications for teaching debugging. To this end, they asked the participants for their reactions in four real-world troubleshooting situations, such as giving instructions to repair a broken light bulb, troubleshooting the popular children's game "telephone", or describing their reactions to further real-world troubleshooting instances from the participants' lives. The subjects (university students) answered in written form. The authors then analyzed those answers for common characteristics and compared them to the debugging behaviors of novices and experts. From the results, they conclude implications for teaching debugging, such as the need to "address the differences between locating an error and fixing it", emphasizing the importance of test-only, or that undoing a previous step is unnatural behavior for students. They conclude that debugging is far less "common sense" than sorting or concurrency is.

In summary, existing research suggests that preexisting debugging experience, in the form of troubleshooting, influences debugging behavior. However, written assessment, which is commonly used within the measurement of preexisting experience, does not allow for observing the actual troubleshooting behavior. A survey with open questionnaires cannot record the students' reactions when their original plan does not work out, and enables the participants to plan or revise their final response comprehensively. Therefore, the relevant characteristics of their troubleshooting behavior cannot be assessed. In contrast, escape rooms as a research method provide a promising approach to study students' actual troubleshooting process in a natural environment.

B. Escape Rooms as Research Method

For some years now, so-called (live) escape rooms (or escape games, exit-the-room games, breakout games, etc.), in which participants are "locked" into a room and have to escape from it, have been widely used by commercial providers who have translated the original idea from a subgenre of digital point-and-click adventures into reality [36]. A typical escape room has an overarching story and the participants have to solve a variety of tasks in a given time. This often includes unlocking corresponding locks or similar, which in turn lead to new puzzles and tasks. To win, all puzzles must be solved in the given time. The players must then either escape from the room and/or find a certain item.

In recent years, there has been a growing interest in the use of such escape rooms in various areas of both informal and formal education. The use of an escape room as a *teaching method* offers various exciting educational possibilities, such as an appealing and motivating context through the application of elements of gamification [37], [38]. In addition, skills such as collaboration, critical thinking, and problem solving can be

fostered in a natural way [39]–[41]. The objectives of such spaces range from teaching general problem-solving or team coordination skills (e.g. [40], [42]) to teaching content that is specific to a certain domain or subject area, such as computer science [41], pharmacy [43], or physics [44].

For many of these escape rooms, which are used as a teaching method, factors such as design criteria, learner motivation, or the actual learning success have been evaluated. However, the potential of escape rooms as a method for investigating problem-solving and learning processes is, in contrast, still largely unused.

Järveläinen and Paavilainen-Mäntymäki, [45] carried out a comparative case study in which they analyzed the learning processes of three student teams in an escape room to convey an information science research method. They found that the different teams used different learning processes on their way through the escape room.

In computer science education, Hacke [41] analyzed behavioral patterns in problem-solving processes. For this purpose, the author examined the video recordings of 38 groups of students. For the analysis, he used a deductive category system to classify behavioral patterns in the problem solving process. Subsequently, the influence of these patterns on the overall success in the game was evaluated. The author identified promising behavioral patterns and group behavior. These include, for example, the use of a blackboard for taking notes, a coordinator (instead of a leader) to coordinate the team in the group composition, or structured task solvers in the team. Most of these patterns, however, are on a rather abstract level of "team composition" or deal with team characteristics rather than being applied more closely to the actual problem solving process.

In summary, the use of escape rooms as a research method represents a – so far – hardly used but promising opportunity to study problem solving processes. It makes it possible to observe the processes in a "natural" environment. This way, the analysis of preexisting debugging experience can be taken one step further than before: Instead of only letting participants describe how they would react and proceed in a given situation, the actual problem solving processes and behavior in a real problem solving situation can be captured.

IV. Development of the Escape Room

This study aims to develop an instrument to assess and analyze students' preexisting debugging experience. To this end, students' behavior in troubleshooting situations has to be investigated. In contrast to existing approaches that used real-world examples of troubleshooting situations to determine preexisting debugging experience, we take this approach a step further: Instead of asking participants for how they would react if they were put into a given situation, we actually put them in that situation by using a real-world escape room setting. Such an escape room approach offers the following potentials for the assessment of preexisting debugging experience:

- The actual troubleshooting process and its characteristics and features can be observed in a "natural environment".
- In contrast to the survey by open questionnaires or interviews, it also records the students' reactions when their original plan does not work out.
- In addition, the participants are not able to plan or revise their final response comprehensively, as would be the case with a written survey.

In designing the escape room as an assessment instrument, the existing experiences from using escape rooms in education described above, both as teaching and scientific survey methods, offer a certain degree of orientation. Nevertheless, any experience gained in terms of research interest and specific scientific objectives can only be transferred to a limited extent. Therefore, a design-based research approach was chosen to develop the escape room and its tasks.

After the objectives of the room had been clearly defined, the first step (see [46]) was to determine the overarching story in which all tasks were to be thematically embedded. The "office of a professor" was chosen as the setting because of the organizational conditions (use of rooms at the university) and because of the variety of possibilities it offers. This story was linked to "ancient Egypt": The goal of the participants is to end the curse of the pharaoh by finding an object stolen from the excavation site in the office of a professor of archaeoinformatics involved in excavations in the Valley of the Kings. In the next step, the individual tasks were developed. Due to the research project, different criteria for the design of the individual tasks were developed.

A. Content-related Criteria

First of all, the content-related criteria are of importance. The investigation aims to observe the troubleshooting process of students in order to draw conclusions about their preexisting debugging experience. Thus, criteria regarding troubleshooting tasks provide guidelines for designing the content. According to Jonassen and Hung [13] those tasks fulfill the following criteria:

They

- are not completely defined,
- require that a mental model of the system to be troubleshot is constructed,
- have well-known solutions with clear success criteria,
- require that a judgment be made about the nature of the problem,
- usually contain only one error, even if this can lead to several observable errors.

In addition, a clear reference to debugging must be established, so that particular troubleshooting skills that correspond to concrete debugging skills can be applied in solving the task. Above all, the tasks needs to force students

- to apply a systematic approach according to the general troubleshooting process and
- to use global strategies to local debugging strategies, such as *functional/discrepancy detection* or *topographic search*

B. Methodological criteria

The application of corresponding behaviors by the participants must also be observable in order to be able to collect empirical data. Therefore the behavior must either

- be directly and externally observable so that direct conclusions can be drawn about the characteristics and features of the troubleshooting process, or
- can be made observable through teamwork and open communication between team members (see e.g. [47]) by actively promoting such communication or making it necessary.

C. Criteria of escape room setting and practical implementation

Additional criteria result from the selected setting of the escape room (instead of, for example, a laboratory situation):

- **Comprehensibility:** Due to the problem solving character of the tasks in an escape room, the comprehensibility of the tasks is central. It is fundamental to the observation of a goal-oriented troubleshooting process that students can grasp the objective of the task in a reasonable time.
- **Practicability:** Since rooms of the university are used for the practical implementation, which are only available for a certain period of time, the tasks must be able to be set up variably and dismantled within a reasonable time frame.
- **Interlinkability:** The solution of the individual tasks must be suitable as "input" for locks, which then release further tasks. Therefore, the solutions should consist of numerical combinations or short alphanumeric character strings.
- **Thematic fit:** The tasks should be thematically embedded in the overall story.
- **Temporal fit:** The complexity of the tasks must be adapted to the chosen time limit.

According to these criteria, 10 tasks were initially developed, tested, and refined in several iterations with other researchers and K12 students. Via a HD surveillance camera mounted in the room, the participants' behavior and audio could be recorded. We used the camera's two-way audio feature to record the participants' communication and to interact with them to control the game if needed.

The observation and evaluation revealed various challenges and potential for improvement in the (further) development of the tasks according to the design-based research methodology, such as regarding the observability of students' behavior, link to debugging, or arrangement and structure of the material. After the respective refinements, the escape room allowed for studying the students' troubleshooting processes, strategies, and behavior in sufficient detail and, in consequence, drawing conclusions for the teaching of debugging.

V. Resulting Escape Room Tasks

In the following, the final tasks that are most meaningful for the research interest will be described. For each task, first a general description is provided. Then the intended research goal as well as the expected troubleshooting behavior is stated. An overview of the tasks and the respective investigation goals can be found in table I.

1) Screen: The participants find a monitor that does not seem to work. To "repair" it, all they have to do is connect the power cable next to the monitor to the monitor. A streaming receiver is also connected to the monitor (and has a "do-not-touch" label on it) so that after the power is connected, the monitor displays a timer with the remaining time and the next task. The goal of this task is to observe the general troubleshooting process and the application of a *functional/discrepancy detection* strategy.

The screen task always posed the first puzzle of the room, as it provided the students with a timer and necessary information for the next task upon completion. We expected the students to:

- notice that the monitor is not working *(formulate problem description)*.
- hypothesize, based on this observation, that no power source is connected *(generate causes)*.
- press the power or other buttons, and/or check the inputs systematically *(test)*.
- search for the respective cable (if not spotted beforehand) and plug it in, solving the task *(repair and evaluate)*.

2) Tangle of cables: The participants find eight daisy-chained USB-cables, which connect an LED flashlight to a power supply. They also find a box permanently mounted in the room, which they cannot see into without the flashlight. Two of the cables are broken. The task is to identify the broken cables and to form a sufficiently long chain with the remaining cables to be able to shine light into the box and read the numbers in it. The goal of this task is to observe the general troubleshooting process and the application of a *functional/discrepancy detection* strategy.

Fig. 2. Tangle of cables

- plug in the daisy-chained USB cables, noticing the flashlight not working *(formulate problem description)*.
- generate multiple hypotheses based on this observation, such as the power socket, the power adapter, the cables, or the flashlight itself not working *(generate causes)*.
- systematically check those hypotheses by testing the respective component (*test* by using strategies such as *functional/discrepancy detection*).

- eventually conclude that some of the cables are not working, identify those cables and build the daisy-chain without them, allowing them to shine light into the box and read the numbers inside *(repair and evaluate)*.

3) Tap the telephone: Participants discover a box with five cables hanging out. When a cable is plugged into one of the five plugs, they receive audio feedback as to how many cables are correctly connected. They must find the correct order of the cables, similar to the game *Mastermind*. The goal of this task is to observe the general troubleshooting process.

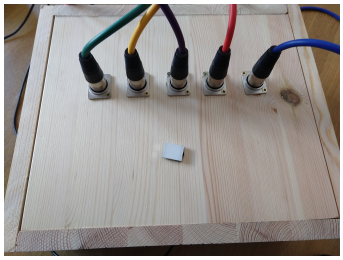

Fig. 3. Tap the telephone

For this task, we expected the students to:

- randomly connect cables and sockets *(formulate problem description)*.
- in doing so, notice a correlation between the audio feedback and the connected wires and, therefore, understand the system *(generate causes/test)*.
- systematically find the correct position for each cable step by step by checking each of the remaining sockets for a cable *(repair and evaluate)*.

4) Valley of the Kings: Participants find a map of the Valley of the Kings with a route on it. They will also find a route description, although some of the arrows describing the route are wrong. The paper with the route contains a note that 6 errors can be identified. This activity is designed to analyze the application of a topographic search strategy.

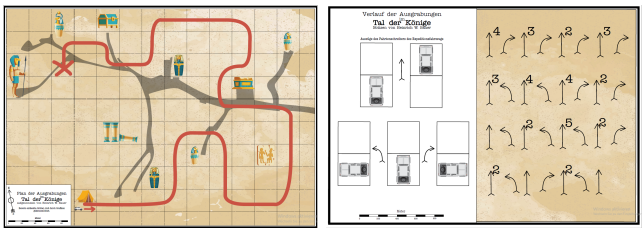

Fig. 4. Valley of the kings

For the valley-of-the-kings-task, we expected the students to:

- understand the system of arrow directions and route using the given example and compare the first few directions with the route *(formulate problem description)*.
- trace the route, using some form of auxiliary material to help keep track of the wrong arrows and/or the current position *(apply forward topographic search strategy)*.

5) Finding Mr. X: A web application (on a previously found tablet) shows a map of the Cairo subway system. The participants have the task to find out where Mr. X, whose starting position is given, is going. To do this, they can place two "watchers" in the subway tunnels and get feedback on whether Mr. X has passed them. After a waiting period of 30 seconds, they will receive another chance to place the guards. This activity was developed to analyze the application of a topographic search strategy.

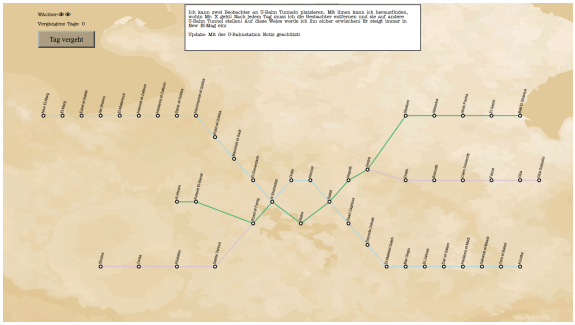

Fig. 5. Finding Mr. X

For this task, we expected the students to:

- start from the given entry subway station
- systematically isolate the exit station by placing watchers that provide as much information as possible regarding the route of Mr. X, such as at tunnels after transit stations *(apply forward topographic search strategy)*.

VI. Discussion

The observation and evaluation revealed various challenges and potential for improvement in the (further) development of the tasks, which were addressed as follows.

1) What is the solution?: Due to the troubleshooting nature of the tasks, students were often confronted with finding errors in a system. A typical hurdle was to recognize whether the *errors* (e. g. incorrect numbers) or their *correction* (e. g. the correct numbers replacing the incorrect ones) represent the combination that is needed, for example, to open the next lock. In order to solve this problem, the *errors* were uniformly defined as the solution needed, since the focus in the majority of tasks was on error localization and not on the correction. This convention was communicated to the participants at the beginning of the game, thus reducing the problem.

2) What do I have to do here?: For some of the tasks the participants had problems understanding the objective of the task without external advice from the game master. Accordingly, additional hints on the material were added or existing ones were improved or elaborated. Furthermore, in the first iterations, many materials for later tasks were already available

TABLE I
OVERVIEW OF THE RESULTING TASKS

Task	**Research Goal**
Screen	general troubleshooting process and application of a *functional/discrepancy detection* strategy
Tangle of cable	general troubleshooting process and the application of a *functional/discrepancy detection* strategy.
Tap the telephone	general troubleshooting process
Valley of the kings	topographic search strategy
Finding Mr. X	topographic search strategy

in the room, in part, because there were no possibilities to lock them away and to release them later on. This caused confusion in a lot of groups. In order to prevent students from trying to establish a connection between materials of different tasks in a time-consuming and unsuccessful manner, further solutions to unlock them step-by-step were developed (for example, by purchasing additional lockable boxes of suitable size).

3) Complexity and Time: In the first iteration, some of the required materials were hidden somewhere in the room, as it is common for the escape-room genre [36]. It turned out that the students had great problems to search and find these items efficiently. Since the search does not contribute to the goal of the investigation, hidden objects were completely omitted and all necessary objects were positioned centrally in the room so that they could be found directly. It was also noticed that the student groups rarely worked in parallel when several tasks were available. Therefore the number and complexity of the tasks were reduced.

4) Bruteforce: Another striking pattern was that many groups of students tried to open individual locks by trying all possible combinations – especially when one or two parts of the number combination were already known. However, this behavior, which was not used in the actual tasks but rather to *bypass* them, could only be prevented to a limited extent, for example by using appropriate locks. Other shortcuts, such as the extraction of materials from locked chests or a diary, were also tried to be prevented by firmly fixing these materials.

In addition to these factors, which are primarily relevant for escape-room setting criteria, methodological and content-related problems were identified as well:

5) Observability of the procedure: For some of the tasks, the students' procedure was only observable to a limited extent, especially because tasks were solved "in the mind" – or at least attempted to. Therefore, in those cases, the troubleshooting procedure and its characteristics as well as applied strategies could not be collected. By providing additional external aids such as colored stones to mark positions, the corresponding processes could be made visible. Furthermore, measures were taken to actively promote communication among the participants in order to make internal processes visible: For example, a time limit was introduced for one task until the next attempt was possible, during which the participants could plan the further procedure.

6) Reference to debugging: In some tasks it also became apparent that no clear connection could be established between the (troubleshooting) actions and debugging. Due to the combination of missing or unclear reference to debugging and problems in observability, some tasks were omitted because they could not contribute to the research goal.

A. Evaluation and Limitations

Overall, the evaluation of the escape room showed that the ability to observe students' behaviors, processes, and strategies were satisfactory. Within the evaluation, we had groups in which one student solved a certain task on his own as well as groups that solved a certain task in such a way that the procedure could not be observed closely due to the camera angle. Consequently, these cases need to be left out of the analysis. Nevertheless, for the vast majority of the tasks and groups, the actions of the students could be observed and additional insights could be gained from group discussions and communication. Furthermore, all groups showed a high level of motivation due to the scenario. Since the room was not linear, participants could sometimes work on certain tasks in parallel. They were thus always able to search the room for further clues for later problems. This may influence their tendency to abandon a certain task and concentrate on another one for the time being (although only a few groups were observed that split up to work on tasks in parallel).

The instrument developed within several iterations of the design-based research process can now be used to analyze students' troubleshooting behavior. As debugging related behaviors like a systematic troubleshooting process or certain global troubleshooting strategies are necessary and observable within the tasks, preexisting debugging experience that influences novices' debugging behavior can be studied and identified. This allows us to draw conclusions for the teaching of debugging in the classroom.

VII. CONCLUSION

In summary, we developed an innovative methodological approach to study students' troubleshooting behavior. The main advantage of this instrument in comparison to a written assessment of participants' reactions in given situations is that we can observe the actual troubleshooting process and strategies in a natural environment, including the reactions that occur if an initial approach does not work out. The communication within the groups, which was actively fostered in the design of the tasks, turned out to make the participants' processes observable.

Due to the limited existing preliminary work, a design-based research approach was chosen. This turned out to be suitable for developing appropriate activities to enable the observation of troubleshooting behavior.

Furthermore, the tasks developed within the design-based research process provide a valuable resource for the classroom. They allow us to point out similarities between troubleshooting and debugging and fostering certain practices such as a systematic process and the generation of hypotheses in a non-programming domain. This way, they might even contribute to helping students employ corresponding debugging skills in their daily lives in the sense of computational thinking.

References

[1] M. Perscheid, B. Siegmund, M. Taeumel, and R. Hirschfeld, "Studying the advancement in debugging practice of professional software developers," *Software Quality Journal*, vol. 25, no. 1, pp. 83–110, 2017, ISSN: 15731367.

[2] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen, "A study of the difficulties of novice programmers," *Acm Sigcse Bulletin*, vol. 37, no. 3, pp. 14–18, 2005.

[3] D. N. Perkins, C. Hancock, R. Hobbs, F. Martin, and R. Simmons, "Conditions of learning in novice programmers," *Journal of Educational Computing Research*, vol. 2, no. 1, pp. 37–55, 1986.

[4] T. Michaeli and R. Romeike, "Current status and perspectives of debugging in the k12 classroom: A qualitative study," in *2019 IEEE Global Engineering Education Conference (EDUCON)*, Dubai, VAE: IEEE, 2019, pp. 1030–1038.

[5] J. Bonar and E. Soloway, "Preprogramming knowledge: A major source of misconceptions in novice programmers," *Human–Computer Interaction*, vol. 1, no. 2, pp. 133–161, 1985.

[6] R. Duit, H. Gropengießer, and U. Kattmann, "Towards science education research that is relevant for improving practice: The model of educational reconstruction," in *Developing standars in research on science education edition*, H. Fischer, Ed., London, UK: Taylor & Francis, 2005, pp. 1–10.

[7] D. H. Jonassen, "Toward a design theory of problem solving," *Educational technology research and development*, vol. 48, no. 4, pp. 63–85, 2000.

[8] A. Yadav, N. Zhou, C. Mayfield, S. Hambrusch, and J. T. Korb, "Introducing Computational Thinking in Education Courses," in *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '11, New York, NY, USA: ACM, 2011, pp. 465–470, ISBN: 978-1-4503-0500-6.

[9] B. Simon, D. Bouvier, T.-Y. Chen, G. Lewandowski, R. McCartney, and K. Sanders, "Common sense computing (episode 4): Debugging," *Computer Science Education*, vol. 18, no. 2, pp. 117–133, 2008.

[10] J. R. Anderson and J. Crawford, *Cognitive psychology and its implications*. San Francisco, CA, USA: Worth Publishers, 1980.

[11] N. M. Morris and W. B. Rouse, "Review and evaluation of empirical research in troubleshooting," *Human factors*, vol. 27, no. 5, pp. 503–530, 1985.

[12] I. Katz and J. Anderson, "Debugging: An analysis of bug-location strategies," *Human-Computer Interaction*, vol. 3, no. 4, pp. 351–399, 1987.

[13] D. H. Jonassen and W. Hung, "Learning to troubleshoot: A new theory-based design architecture," *Educational Psychology Review*, vol. 18, no. 1, pp. 77–114, 2006.

[14] A. Schaafstal, J. M. Schraagen, and M. Van Berl, "Cognitive task analysis and innovation of training: The case of structured troubleshooting," *Human factors*, vol. 42, no. 1, pp. 75–86, 2000.

[15] G. Klein, "Recognition-primed decisions," *Advances in man-machine system research*, vol. 5, pp. 47–92, 1989.

[16] D. Dounas-Frazer, K. Van De Bogart, M. Stetzer, and H. Lewandowski, "Investigating the role of model-based reasoning while troubleshooting an electric circuit," *Physical Review Physics Education Research*, vol. 12, pp. 010137-01 –010137-20, 1 2016.

[17] J. Rasmussen and A. Jensen, "Mental procedures in real-life tasks: A case study of electronic trouble shooting," *Ergonomics*, vol. 17, no. 3, pp. 293–307, 1974.

[18] N. Reed and P. Johnson, "Analysis of expert reasoning in hardware diagnosis," *International Journal of Man-Machine Studies*, vol. 38, no. 2, pp. 251–280, 1993.

[19] S. Bereiter and S. Miller, "A field-based study of troubleshooting in computer-controlled manufacturing systems," *IEEE transactions on Systems, Man, and Cybernetics*, vol. 19, no. 2, pp. 205–219, 1989.

[20] U. Konradt, "Strategies of failure diagnosis in computer-controlled manufacturing systems: Empirical analysis and implications for the design of adaptive decision support systems," *International journal of human-computer studies*, vol. 43, no. 4, pp. 503–521, 1995.

[21] C. Li, E. Chan, P. Denny, A. Luxton-Reilly, and E. Tempero, "Towards a framework for teaching debugging," in *Proceedings of the Twenty-First Australasian Computing Education Conference*, New York, NY, USA: ACM, 2019, pp. 79–86.

[22] J. D. Gould, "Some psychological evidence on how people debug computer programs," *International Journal of Man-Machine Studies*, vol. 7, no. 2, pp. 151–182, 1975.

[23] D. Spinellis, "Modern debugging: The art of finding a needle in a haystack," *Communications of the ACM*, vol. 61, no. 11, pp. 124–134, 2018.

[24] B.-d. Yoon and O. Garcia, "Cognitive activities and support in debugging," in *Proceedings Fourth Annual Symposium on Human Interaction with Complex Systems*, Dayton, OH, USA: IEEE, 1998, pp. 160–169.

[25] M. Guzdial, "Learner-centered design of computing education: Research on computing for everyone," *Synthesis Lectures on Human-Centered Informatics*, vol. 8, no. 6, pp. 1–165, 2015.

[26] D. Klahr and S. Carver, "Cognitive objectives in a logo debugging curriculum: Instruction, learning, and transfer," *Cognitive Psychology*, vol. 20, no. 3, pp. 362–404, 1988.

[27] D. C. Phillips, "The good, the bad, and the ugly: The many faces of constructivism," *Educational researcher*, vol. 24, no. 7, pp. 5–12, 1995.

[28] J. Bransford, A. Brown, and R. Cocking, *How people learn*. Washington DC, USA: National Academy Press, 2000.

[29] L. Onorato and R. W. Schvaneveldt, "Programmer-nonprogrammer differences in specifying procedures to people and computers," *Journal of Systems and Software*, vol. 7, no. 4, pp. 357–369, 1987.

[30] L. A. Miller, "Natural language programming: Styles, strategies, and contrasts," *IBM Systems Journal*, vol. 20, no. 2, pp. 184–215, 1981.

[31] P. Gibson and J. O'Kelly, "Software engineering as a model of understanding for learning and problem solving," in *Proceedings of the first international workshop on Computing education research*, New York, NY, USA: ACM, 2005, pp. 87–97.

[32] Y. B.-D. Kolikant, "Gardeners and cinema tickets: High school students' preconceptions of concurrency," *Computer Science Education*, vol. 11, no. 3, pp. 221–245, 2001.

[33] B. Simon, T.-Y. Chen, G. Lewandowski, R. McCartney, and K. Sanders, "Commonsense computing: What students know before we teach (episode 1: Sorting)," in *Proceedings of the second international workshop on Computing education research*, New York, NY, USA: ACM, 2006, pp. 29–40.

[34] G. Lewandowski, D. Bouvier, R. McCartney, K. Sanders, and B. Simon, "Commonsense computing (episode 3) concurrency and concert tickets," in *Proceedings of the third international workshop on Computing education research*, New York, NY, USA: ACM, 2007, pp. 133–144.

[35] T. VanDeGrift, D. Bouvier, T.-Y. Chen, G. Lewandowski, R. McCartney, and B. Simon, "Commonsense computing (episode 6) logic is harder than pie," in *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, New York, NY, USA: ACM, 2010, pp. 76–85.

[36] S. Nicholson, "Peeking behind the locked door: A survey of escape room facilities," Wilfrid Laurier University, Tech. Rep., 2015.

[37] C. Borrego, C. Fernández, I. Blanes, and S. Robles, "Room escape at class: Escape games activities to facilitate the motivation and learning in computer science," *JOTSE*, vol. 7, no. 2, pp. 162–171, 2017.

[38] S. Nicholson, "Creating engaging escape rooms for the classroom," *Childhood Education*, vol. 94, no. 1, pp. 44–49, 2018.

[39] R. Pan, H. Lo, and C. Neustaedter, "Collaboration, awareness, and communication in real-life escape rooms," in *Proceedings of the 2017 Conference on Designing Interactive Systems*, ACM, New York, NY, USA, 2017, pp. 1353–1364.

[40] C. Friedrich, H. Teaford, A. Taubenheim, P. Boland, and B. Sick, "Escaping the professional silo: An escape room implemented in an interprofessional education curriculum," *Journal of interprofessional care*, vol. 33, no. 5, pp. 573–575, 2019.

[41] A. Hacke, "Computer science problem solving in the escape game "room-x"," in *Informatics in Schools. New Ideas in School Informatics*, S. N. Pozdniakov and V. Dagienė, Eds., Cham: Springer International Publishing, 2019, pp. 281–292, ISBN: 978-3-030-33759-9.

[42] P. Williams, "Using escape room-like puzzles to teach undergraduate students effective and efficient group process skills," in *2018 IEEE Integrated STEM Education Conference (ISEC)*, Princeton, NJ, USA: IEEE, 2018, pp. 254–257.

[43] H. Eukel, J. Frenzel, and D. Cernusca, "Educational gaming for pharmacy students–design and evaluation of a diabetes-themed escape room," *American journal of pharmaceutical education*, vol. 81, no. 7, p. 6265, 2017.

[44] A. I. V. Vörös and Z. Sárközi, "Physics escape room as an educational tool," in *AIP Conference Proceedings*, AIP Publishing, vol. 1916, 2017, p. 050 002.

[45] J. Järveläinen and E. Paavilainen-Mäntymäki, "Escape room as game-based learning process: Causation-effectuation perspective," in *Proceedings of the 52nd Hawaii International Conference on System Sciences*, Waikoloa Village, HI, USA: ScholarSpace 2019, 2019.

[46] S. Clarke, D. Peel, S. Arnab, L. Morini, H. Keegan, and O. Wood, "Escaped: A framework for creating educational escape rooms and interactive games for higher/further education," *International Journal of Serious Games*, vol. 4, no. 3, pp. 73–86, 2017.

[47] D. A. Fields, K. A. Searle, and Y. B. Kafai, "Deconstruction kits for learning: Students' collaborative debugging of electronic textile designs," in *Proceedings of the 6th Annual Conference on Creativity and Fabrication in Education*, New York, NY, USA: ACM, 2016, pp. 82–85.