International Education Studies; Vol. 11, No. 12; 2018
ISSN 1913-9020 E-ISSN 1913-9039
Published by Canadian Center of Science and Education

Do Computer Science Students Understand Their Programming Task?—A Case Study of Solving the Josephus Variant Problem

Andreas Febrian[1] & Oenardi Lawanto[1]

[1] Engineering Education Department, College of Engineering, Utah State University, Utah, USA

Correspondence: Andreas Febrian, Engineering Education Department, College of Engineering, Utah State University, Utah, USA. E-mail: andreas.febrian@gmail.com

Received: July 18, 2018 Accepted: September 1, 2018 Online Published: November 27, 2018

doi:10.5539/ies.v11n12p26 URL: https://doi.org/10.5539/ies.v11n12p26

Abstract

The ability of students to problem solve begins with interpreting the problem. When they interpret the problem inaccurately, they will likely use ineffective strategies or fail to solve the problem. Studies reported students are often incapable of identifying and articulating the problem goal, requirements/constraints, and expected output. In other words, students lack self-regulation skills, especially related to task understanding. In this study, two male and two female senior computer science students from Utah State University, USA, were recruited as research participants to learn more about their task understanding skills while engage in programming tasks. The participants were asked to answer five programming problems while thinking aloud, and their responses were video- and audio-recorded. This report focuses on one of the problems, which was a variant of the Josephus problem. Three research questions were used to guide the analysis: (a) what were the participants' initial task understanding; (b) how did it change during the problem-solving endeavor; and (c) why did it change. All participants identified the problem goal inaccurately and as a result, selected ineffective problem-solving strategies. The analysis suggested their inaccurate task interpretations were caused by their confidence bias (i.e., a systematic cognitive error), in which they drew knowledge and strategies from irrelevant experience. Out of four participants, only one was able to defeat the confidence bias and acquired an accurate task understanding; the influencing factors and possible interventions to overcome confidence bias are discussed.

Keywords: programming, task interpretation, task understanding, self-regulated learning, problem-solving, task revision, cognition

1. Introduction

Numerous technologies seamlessly support our daily activities in this digital age. Presently, job automation and technology integrated solutions are becoming more common (Bui, 2015; Hambrusch, Hoffmann, Korb, Haugan, & Hosking, 2009; Henderson, 2009), and the demand for competent computer scientists is increasing (Lacey & Wright, 2009). Unfortunately, many undergraduate students are struggling to adapt to the expectations of computer science (CS) education. Various CS departments reported 30% to 50% dropout rate of undergraduate students (AAA Utah State University, 2016; Beaubouef & Mason, 2005; Howles, 2007; Kori et al., 2015). While many factors may influence students' decision to leave their major, numerous studies reported the immense challenges in learning computer programming during the first year as the leading dropout reason (Anderson & Skwarecki, 1989; Guzdial et al., 2015; Howles, 2007; Kori et al., 2015).

Computer programming plays a critical role in the CS discipline (Denning, 2003), in that it is believed to be the most effective and efficient way to learn various CS concepts and principles (Gal-Ezer & Harel, 1998; Lye & Koh, 2014; Wing, 2006, 2008). Unfortunately, instructing students to learn programming is challenging. Gal-Ezer & Harel (1998) argues that some programming concepts are hard to teach and be absorbed by students, such as recursion and that a program is rigid "yet is supposed to deal with many different inputs of varying sizes" (p.83). Students also encounter learning impediment due to having incorrect perspectives of the computer (Ben-Ari, 1998; Brennan & Resnick, 2012; Lischner, 2001), using inefficient learning strategies (Whittington, 2004), or facing an undesirable learning environment (e.g., lack of human interaction) (Ben-Ari, 1998). Consequently, numerous researchers attempted to tackle these issues by developing various computer-based instructional tools and enhancing existing instructional methods (Adams, 2007; Briggs, 2005; Gonzalez, 2006; Krauss, 2008; Lonchamp, 2010; Lu, Conley, Klein, & Drive, 2014; Resnick et al., 2009; Rum & Ismail, 2017; Ruthmann, Heines, Greher,

Laidler, & Saulters, 2010; X.-M. Wang, Hwang, Liang, & Wang, 2017; Whittington, 2004; Williams, Wiebe, Yang, Ferzli, & Miller, 2010).

Students' learning impediment and high dropout rate can also be caused by their inadequate self-regulation (SR) skills. Falkner, Vivian, & Falkner (2014) reported some students are incapable of considering and addressing the assessment criteria during their problem-solving enterprise. Peng, Wang, & Sampson (2017) reported students are unable to identify the problem goal and expected output correctly. Although these reports suggest a deficiency in SR strategies, it is still unclear which SR aspect the students lack. Nevertheless, out of various SR aspects (e.g., knowledge about self, task interpretation, or planning), task interpretation or understanding of the problem is the most critical because it shapes the rest of the problem-solving endeavor (Butler & Cartier, 2005). Thus, students' self-regulation, especially their task interpretation became the focus of this study.

2. Literature Review

2.1 Self-Regulation and Task Interpretation

Self-regulation (SR) is defined as a complex, situated, and iterative goal-directed activity (Butler & Cartier, 2005; Butler, Schnellert, & MacNeil, 2015; Butler & Winne, 1995). All SR activities are intentional, deliberate, and aimed to achieve desired goals (Butler, Schnellert, & Perry, 2017; Zimmerman, 2008). When self-regulating, students are also affected by their emotions and motivations (Butler & Cartier, 2005; Zimmerman, Heart, & Mellins, 1989). Students who are competent with using various SR skills tend to be more successful academically (Butler & Cartier, 2005; Coutinho, 2007), and to be better engineers (Lawanto et al., 2013). Also, students' SR is influenced by their peers (Hadwin, Jarvela, & Miller, 2011; Rivera-Reyes, Lawanto, & Pate, 2016), in that one student might help other students improve their SR (i.e., co-regulation) or a group of students might develop particular SR strategies (i.e., shared-regulation).

Since 1996, various researchers, such as Zimmerman, Winne, Hadwin, Pintrich, Butler, and Cartier, have tried to capture the complexity of students' SR in a single framework (Santoso, 2013). Butler and Cartier's model (BCM) of SR was chosen for the study presented in this paper because it emphasizes the significance of various contexts in any regulatory activities. Precisely, the BCM imposes that all self-regulation activities are influenced by multiple layers of context (Butler & Cartier, 2004, 2005; Butler et al., 2017; Cartier & Butler, 2004), such as students' awareness of themselves (e.g., their experience or strengths), academic culture, computer science (CS) as a discipline, and concepts, standards, and best practices in programming. This contextual characteristic makes BCM flexible because it can be and has been used to describe CS and engineering students' SR (Febrian, Lawanto, & Cromwell, 2015; Lawanto, 2010; Lawanto et al., 2013; Lawanto, Butler, Cartier, Santoso, & Goodridge, 2013; Santoso, 2013).

The BCM describes three elements of SR including the multiple layers of context, the learning or problem-solving environment, and students' engagement with the environment (Butler & Cartier, 2004, 2005; Butler et al., 2017; Cartier & Butler, 2004). The problem-solving environment refers to the given task, available resources and supports, and provided feedback. Students' engagement with the learning environment refers to their activities of (1) understanding or interpreting the task, (2) developing a plan to solve the task, (3) enacting the plan, (4) monitoring their progress and problem-solving activities, and (5) making adjustment as necessary. Adjustments can occur in their task interpretation (TI), plan, or enactment strategies based on their monitoring results. Additionally, the cycle of strategic action taken by a student (i.e., TI, planning, enacting, monitoring, and adjusting) is influenced by emotions and motivations. Butler and Cartier (2004, 2005), Butler et al. (2017), and Cartier and Butler (2004) provide a more detailed discussion of BCM.

Butler (1998) describes students' TI as their understanding of the given task and the associated processing demand to complete it. However, this definition is too broad to be operationalized. Thus, this study complemented the BCM with Hadwin's model of TI. Hadwin (2006) argues students' understanding of a problem can be categorized as explicit or implicit TI. Explicit TI refers to information that is plainly presented in a task description (Hadwin, Oshige, Miller, & Wild, 2009), such as the task goal. Implicit TI refers to any extrapolated information from the task description (Hadwin et al., 2009), such as relevant concepts, knowledge, and experience. These definitions imply that if the goal needed to be extrapolated from the task description, then it was part of the implicit TI. Likewise, if the task description listed the relevant concepts, then it was part of the explicit TI. In other words, how students get the information (i.e., by identifying or extrapolating) determines how the information is categorized.

Task interpretation is the first step in any SR activity (Butler & Cartier, 2005). Thus, any inaccurate explicit or implicit TI may inform students to select and employ ineffective problem-solving approaches (Butler, 1995). Fortunately, TI changes during the problem-solving enterprise (Rivera-Reyes, 2015; Rivera-Reyes, Lawanto, &

Pate, 2017), suggesting that even though students may start with inaccurate explicit and implicit TI, they may gain the correct interpretation while solving the task, providing they continuously monitor themselves and are open to necessary adjustments. Isomöttönen & Tirronen (2013) argues that having relevant knowledge and skills on the given task is necessary for accurate and efficient self-monitoring activities. Further, when students are familiar with the discipline-related knowledge, values, skills, and expertise, they tend to have more accurate TI and be more competent of selecting effective domain-specific strategies (Butler & Winne, 1995; Hadwin et al., 2009). In other words, SR skills have a positive influence toward students' problem-solving endeavor (Lawanto, 2010; Lawanto & Johnson, 2009; Pintrich, 2002), and insufficient SR may endanger their problem-solving effort (Schoenfeld, 1983).

2.2 Self-Regulation in Computer Programming

Developing a computer program requires the ability to translate and model one's way of thinking, the problem, and the solution in natural language into the selected programming language (Renumol, Janakiram, & Jayaprakash, 2010). Each programming language has its unique syntax and structures that help students organize their code and focus their attention on solving the problem (Lee, 2014). Further, students use various cognitive skills to comprehend, design, and interlink various types and levels of abstraction while programming (Renumol et al., 2010; Wing, 2008). Self-regulation is therefore essential during such complex endeavor. Unfortunately, there is limited literature on this topic in CS education.

Alharbi, Henskens, & Hannaford (2012) reported typical CS students are visual, sensing, reflective, and sequential learners, suggesting that they learn best when exposed to visual representation and facts sequentially. CS students may also prefer to solve problems sequentially and to develop a visual interpretation of the problem. As reflective learners, CS students may frequently monitor and reflect on their learning progress (Felder & Soloman, n.d.). Lye & Koh (2014) argue that reflections can help students better understand various CS concepts and principles. In other words, most CS students are equipped with some characteristics that could help them be successful in this discipline.

Kumar et al. (2005) reported that using various SR skills enhances students' performance in programming. Since using discipline-specific strategies during a problem-solving endeavor is more efficient (Falkner et al., 2014), it is possible that self-regulated students are more competent in selecting and applying CS discipline-specific strategies, thus enhancing their performance. Bergin, Reilly, & Traynor (2005) reported that students with high intrinsic motivations and task value (i.e., able to appreciate the importance of a given task) tend to employ various SR strategies more frequently, and possibly perform better in programming, compared to their counterparts.

Studies also discussed SR deficiencies of CS students. Havenga (2015) reported that although students are aware of their fragmented knowledge of, and potential misconceptions on, various object-oriented programming concepts, they tend to jump into solving a given task instead of addressing their fragmented knowledge and confirming the potential misconceptions. This report suggests that some CS students are incapable of self-regulating themselves properly. Falkner et al., (2014) reported some CS students are incapable of considering and addressing parts of the assessment criteria during their programming endeavor, suggesting an inadequacy of students' TI skills.

3. Research Questions and Design

The objective of this qualitative study was to investigate the task understanding skills of four senior CS students solving a variant of the Josephus problem. Three research questions were used to guide this study: (1) what were the students' initial explicit and implicit TIs; (2) how did their initial TI change during the problem-solving; and (3) what were the factors that influenced those changes. Multiple qualitative case study design was used to answer the research questions because of its alignment with the golden standard for conducting SR research (Butler & Cartier, 2005, 2018; Dinsmore, Alexander, & Loughlin, 2008) where the researchers collected multiple, in-depth data of the participants' (or cases') SR activities while solving programming problems. This qualitative study consisted of two units of analysis, which were related to object-oriented and algorithm design. Both issues require the participants to work on various levels of abstraction. In this report, the researchers only focus on the algorithm design; the object-oriented design is discussed elsewhere.

3.1 The Participants

In selecting the participants, Creswell (2012) argues that the cases need to represent as much diversity as possible. To establish diversity in this study, two male and two female senior CS students that represented higher- and lower-performance based on GPA were selected. Male and female students were selected based on studies that reported that males and females think, perceive, and self-regulate differently (Irani, 2004; Lawanto, Cromwell, & Febrian, 2016; Madigan, Goodfellow, & Stone, 2007; Pivkina, Pontelli, Jensen, & Haebe, 2009). Higher and lower

performing students were selected because a higher GPA might reflect competent self-regulated students and vice versa, lower GPA might reflect incompetent self-regulated students. This interpretation was based on reports by Butler and Cartier (2005), Coutinho (2007), and Dent and Koenka (2016) that suggest competent self-regulated students tend to have excellent academic achievements.

Two male and two female senior CS students at Utah State University, USA consented to participate in this study. Each of the four participants selected an alias to be used in this study. The higher-performer participants were Jake and Anne with GPAs of 3.96 and 3.62 on a 4-point scale, respectively. The lower-performer participants with GPAs of 3.10 and 3.36 on a 4-point scale were Rusty and LStew. Although all participants were academically successful, they used relatively diverse problem-solving and self-regulation approaches, as expected in this study.

All participants were Caucasian and familiar with imperative and object-oriented programming paradigm. The male participants were also familiar with logic programming. As seniors, they had completed various CS core courses, such as introduction to programming, algorithm and data structure, calculus, discrete mathematics, software engineering, and event-driven programming. Jake, Anne, and LStew were taking the Advanced Algorithms course. Thus, they were equipped with more than the required knowledge to solve the Josephus problem. Further, Jake, Rusty, Anne, and LStew spent around 5800, 4160, 2000, and 2100 hours, respectively, developing their programming skills, suggesting they had ample time to hone their skills.

As women, Anne and LStew struggled with the CS stereotype, and a sense of not belonging, especially prior to their senior year. This phenomenon aligns with various reports on women in computer science (Falkner, Szabo, Michell, Szorenyi, & Thyer, 2015; Graham & Latulipe, 2003; Irani, 2004; Lewis, Anderson, & Yasuhara, 2016; Outlay, Platt, & Conroy, 2017; Wang, Hejazi Moghadam, & Tiffany-Morales, 2017). Lewis, Anderson, and Yasuhara (2016) describes this stereotype as "singularly focused on CS, asocial, competitive, and male" (p.30). Both Anne and LStew often felt discouraged when comparing themselves with their peers. Anne shared that she frequently considered quitting her study, and was concerned with the impostor syndrome, which made her think:

"I know I am not as good as other people think I am, and as soon as they find out how bad I am at programming, then they will realize that I should not be here."

LStew reminisced how the self-comparison game tended to lower her self-esteem and impacted her programming performance. Fortunately, both of them overcame this challenge in their senior year. Anne said during the interview, "For the last four years, I thought that I am not as smart as you guys [her peers] but that was all made up in my head." She further explained, "Because there are not as many women [in CS], you do not have as many people to gauge it off … It is harder to know where you really stand with people."

Each participant completed all the research tasks and received a $40 gift card and a personalized SRL report as a token of appreciation. The personalized reports were written with minimum jargon and described each participants' strengths and weaknesses and some strategies to improve their task interpretation skills. All participants responded positively towards their report.

3.2 Data Collection Method

Four instruments were used in this study, including the initial TI survey, five programming problems, five problem-space maps, and semi-structured interview questions. All instruments had been pilot-tested and refined accordingly prior to the data collection. The initial TI survey consisted of five open-ended questions assessing the participants' initial explicit and implicit task understanding, such as their understanding of the task goal and the required steps to solve it. The five programming problems consisted of two practice questions, one object-oriented question (i.e., a unit of analysis), one break question, and an algorithm design question (i.e., another unit of analysis). The problem-space maps were used to track the participants' problem-solving endeavor, such as completed design issues and design changes. This technique was adopted from Johnson (2008) to enhance the observation quality of the participants' thought processes. The interview questions were used to confirm the researchers' observations during the problem-solving enterprise. For example, the researchers asked one participant to explain his justification of switching from using LinkedList to Array to store rebels-related information, and based on that justification, the researchers decided whether a revised TI influenced the design change or not. The semi-structured interview was selected because it enabled the researcher to adjust the interview process to further explore a particular phenomenon of the participants' self-regulation activities.

During the data collection, the participants had to (1) read the problem out loud; (2) complete the initial TI survey; (3) answer the problem out loud; and (4) answer the interview questions. There was no time limit set for solving the problems. Thinking aloud is commonly accepted as one of the best methods to assess how students think (Bainbridge & Sanderson, 2005). Unfortunately, this method is not without shortcoming. Chi, De Leeuw, Chiu, &

Lavancher (1994) argues that thinking aloud might help the participants better self-regulate, thus affecting this research. Readers are advised to consider this limitation when interpreting the findings.

The participants' primary goal was to answer all given programming problems correctly. All participants had to repeat this procedure for each question, and were video- and audio-recorded. The first and second programming problems were used to familiarize them with the procedure. The data were collected in a well-illuminated conference room that could reduce distractions from curious passersby. Unfortunately, the room's automatic motion sensor was sometimes unable to detect the researchers and the participants, which resulted in the lights turning off, which slightly distracted the participants. Each participant was provided with a pen, pencil, highlighter, white paper, two water bottles, a can of soda, and two chocolate bars. A camera and two voice recorders were placed in front of and beside the participants, respectively. Recorded videos and audios of their problem-solving endeavor, initial TI survey responses, interview responses, problem-space maps, and design artifacts were collected from each participant.

3.3 The Josephus Problem

The Josephus problem is one of the famous computer science puzzles, and has been widely used in programming courses and contests. Josephus was a mathematician who belonged to a rebel group against the Roman Empire. In their last raid, the rebel group was overpowered and trapped by the Romans. The rebels knew that a painful execution was waiting for them if captured, so they came up with an outlandish suicidal method. They threw away all swords except one and gave it to the northmost person. Then they did the following: (a) the person with the sword killed the person on his left; (b) that person passed the sword to the next living person on his left; and (c) repeated all steps until there was only one person standing. Using his expertise in mathematics, Josephus aimed to be the last man standing. For this study, the participants had to simulate each step and then determine Josephus' position. Figure 1 presents an illustration of the expected program's outputs. The number of rebels was between 3 to 40 people.

```
Number of people in the group: 5
1, 2, 3, 4, 5
3, 4, 5, 1
5, 1, 3
3, 5
3
```

Figure 1. Expected program input and output

There were at least five issues that needed to be considered to solve this problem, including reading the number of rebels given by a user, representing the rebels in the program, tracking the sword position, simulating the suicidal method, and displaying the program state each time a rebel perished. According to Bloom's Taxonomy, this problem was at level six, more specifically about creating a product for a specific purpose (Gronlund, Gronlund, & Waugh, 2013). Further, according to Jonassen (2000, 2004, 2010), this problem was a design task since it aimed to produce an artifact, had some undefined criteria, was a situated problem, and had multiple solutions. It was possible to answer this problem using imperative or object-oriented programming paradigm, and each paradigm had many solution variations.

3.4 Data Analysis Method

The recorded videos were transcribed to reduce their complexity (Lapadat & Lindsay, 1999) for the qualitative coding process. The verbatim technique was used during the transcription to minimize data loss. This technique implies every spoken word, including shutters and false starts, must be written as is (Tigerfish, n.d.). Further, the transcription also captured and clarified the participants' activities (e.g., writing something), quick change of issue, and programming concepts by using box brackets, dash ("-"), and capitalizing the first letter, respectively. For example, one of the participants said, "So for int C, C less than size, C plus plus, we set Array equal to new Array-or new Array equal to old Array; [writing it down] Array box C equal to the old Array."

The resulting Josephus-related transcriptions were then qualitatively coded by two experts based on strategic actions of BCM. During this process, Table 1 was used as a coding table. Three transcriptions were coded by one of the researchers and a Ph.D. candidate in engineering education, and one transcription was coded by the same researcher and an information technologist; all experts had experience in imperative programming paradigm. The participating researcher had bachelor and master degrees in computer science. Each expert coded individually and

then met to address their differences. During the discussion, the experts also used the recorded videos and other collected data to resolve their disagreements. The experts produced 732 codes with a Kappa score of 1.00 for each Josephus-related transcription, suggesting a perfect agreement among coders (Viera & Garrett, 2005).

The initial TI survey responses were used to address the first research question, which was assessing the participants' initial TI. However, since the participants answered the survey questions in writing, some of their responses lacked details and contexts. Fortunately, the recorded videos/audios of their verbalization could complement the deficiencies of their written responses. Also, the participants' interview responses were checked in case they forgot to report some of their thought processes when answering the survey.

Table 1. Strategic actions of self-regulation

Strategic Action	Definition
Task interpretation (TI)	Students' understanding of a task and the associated processing demand to complete it (Butler, 1998).
Planning strategies (PS)	Selecting appropriate strategies to complete the task (Butler & Cartier, 2005).
Enacting strategies (ES)	Students' cognitive activities employed as they engage in their work executing the design tasks, as planned, monitored, and adjusted through metacognitive activity (Lawanto, Butler, Cartier, Santoso, Goodridge, et al., 2013).
Monitoring (M)	Students' activities of self-monitor progress, goals, plans, or strategies (adjusting approaches to learning) (Butler & Cartier, 2005).
Adjusting (A)	Students' activities of adjusting goals, plans, or strategies based on self-perceptions of progress or feedback (Butler & Cartier, 2005).

The second research question, which was about the participants' initial TI revision, was answered by analyzing the coded-transcriptions and determining any changes in their initial TI. For each identified revision, the researchers traced it back to associated initial interpretation and examined the associated codes along the trace-path.

To address the third research question, which was identifying the factors that influenced these changes, all collected data and other analysis results were used. The influencing factors were mostly inferred from the coded-transcription by using the trace-path. Some other influencing factors were discovered in the participants' interview responses and recorded video of their initial task interpretation survey responses.

4. Findings

As previously discussed, this paper focuses on the results of the Josephus problem, one of the five programming problems provided to the participants, and the discussion is organized by the three research questions. Table 2 shows the participants' overall performance. As shown in this table, all participants started with an incomplete TI. While LStew failed to submit any solution due to facing challenges during the problem-solving endeavor, the other participants submitted inaccurate solutions. During the problem-solving enterprise, only Rusty was able to change his task understanding and acquire an accurate task interpretation.

Table 2. The participants' overall performance

Performance Aspect	Jake	Anne	Rusty	LStew
Initial task understanding	Incomplete	Incomplete	Incomplete	Incomplete
Task interpretation during the problem-solving	Intact	Intact	Changed	Intact
Final task understanding	Incomplete	Incomplete	Incomplete	Incomplete
Submitted solution	Inaccurate	Inaccurate	Inaccurate	-

4.1 Answering Research Question 1: What Was the Participants' Initial Explicit and Implicit Task Interpretation?

The explicit aspects of the Josephus problem were related to the goal and provided requirements/constraints. Its implicit aspects were related to relevant programming concepts, extrapolated requirements/constraints, and the participants' relevant experience and steps to solve it.

All four participants had an incomplete initial understanding of the problem goal. LStew's understanding represents the other participants' interpretation, in which she thought the goal was "to write pseudocode that figures out what position Josephus should be at, in order to survive." This understanding was incomplete because they were not cognizant to print the pseudocode's state each time a rebel perished. Later, this inaccurate

understanding led to other wrong interpretations and the section of ineffective approaches.

LStew identified the provided requirements/constraints as the pseudocode would "take an input, run through the formula, and return the output." The formula referred to a simplified mathematical model of the provided suicidal method. Rusty also had a similar interpretation, saying, "The algorithm must return the correct position." Both interpretations suggested the pseudocode did not have to print its states. These understandings were influenced by their incomplete interpretation of the task goal. Encouragingly, LStew also accurately identified "there will never be an input of zero or one because then the problem would not exist." Likewise, Rusty was capable of identifying "the chosen number cannot die."

While Rusty and LStew focused on the provided requirements/constraints, Jake's and Anne's interpretation centered on extrapolated information. Since the problem asked for pseudocode, Jake concluded it was unnecessary to think about the speed and memory used by the algorithm. However, he acknowledged the algorithm should be mathematically correct, in that the algorithm should generate the correct output for every given input. As if complementing the others' interpretations, Anne deduced the number of people is "given with function call." Both Jake's and Anne's extrapolated requirements/constraints were correct and supported the idea that an incomplete task interpretation of an aspect of a problem might not harm all follow-up task interpretation activities.

Related to the relevant programming concepts, all participants understood that having basic programming knowledge was necessary to answer the given problem. Among the participants, Anne did not explicitly mention this qualification although she was aware of it, which suggested two interpretations. First, the nature of the Josephus problem implied the need to have programming skills, and thus it was unnecessary to state it plainly. Second, she was not mindful of her implicit connection between the problem and basic programming knowledge. Unfortunately, the researchers did not confirm either possible interpretation. On the other hand, Rusty specified he especially needed the skills of using "Arrays with conditional operators and if statements." Further, Jake and LStew added the necessity to have a competency in "making algorithms out of behaviors." Anne even considered "creative problem-solving" skills as part of the needed qualifications. These observed behaviors suggested the participants were competent in identifying relevant programming concepts to solve the Josephus problem. The finding also supported the idea that having an inaccurate explicit TI might not negatively influence the implicit task understanding.

Jake, Rusty, Anne, and LStew mentioned solving discrete mathematics course assignments as relevant experience to solve the Josephus problem. Interestingly, Rusty did not mention this course during his initial TI, but he did mention it during the interview. However, Rusty's previously discussed problem-solving steps were informed by it, indicating he was unconsciously drawing strategies from that experience. On the other hand, Jake, Anne, and LStew shared they had worked on the "math proof of this problem [but] with a twist" during the course final examination. Naturally, their problem-solving steps were informed by this experience.

All participants' approaches to solving the problem were relatively similar, in that they needed to (1) determine a working pattern by simulating the provided suicidal method manually; (2) program the solution based on the identified pattern; and (3) verify the solution's accuracy. Although she enacted it, LStew did not mention these steps during her initial TI, which suggested she was aware of her problem-solving approach but failed to report it. Plausibly, she often solved similar problems using a similar strategy and was able to retrieve and select it instantaneously when working on the Josephus problem while also being oblivious about it. Such phenomenon is known as tacit expertise and can be acquired through continuous practices (Johnson, 2008).

Two issues emerged from the participants' approaches. First, the participants determined they needed to find a pattern. Rusty and LStew further elaborated the pattern for odd and even number of people might be different. However, finding a working pattern was an unnecessary step since they could easily convert the provided algorithm into pseudocode. Second, all participants assumed a working pattern could be identified after simulating some of the possible inputs manually. Consequently, with the exception of Jake, none of the other participants considered looking for alternative approaches during their initial problem-solving steps, which was a worrisome finding.

Studies reported students tend to solve a problem intuitively and then analytically (Abdillah, Nusantara, Subanj, Susanto, & Abadyo, 2016; Ball, Ormerod, & Morley, 2004; Kahneman, 2003). Further, the learner's experience influences their self-regulation (Butler & Cartier, 2004, 2005; Butler et al., 2015; Cartier & Butler, 2004). Thus, it was possible the participants' discrete mathematics experience significantly influenced their intuition and was reflected in their interpretation of the task goal and problem-solving approach. One of the typical discrete mathematics problems asks students to analyze (e.g., find a pattern) a number sequence and find a simple and succinct formula that could correctly generate the sequence. Plausibly, the participants relied on their experience of

solving such problems. This analysis suggested drawing from irrelevant experience might harm the participants' TI. This finding was aligned with Butler's (1995) argument that students' may select ineffective strategies due to their misinterpretation.

When being asked about his wrong interpretation of the problem during the interview, Rusty explained he might have had "a little bit of overconfidence in thinking that I understood the problem," especially since he had "experience with the problem that I thought was similar but turned out to be very different." Similarly, after reading her personalized report, Anne said, "I got excited at the thought that I would have some background in solving that particular problem." Although studies reported that overconfidence is a common trait among students (Mata, Ferreira, & Sherman, 2013; Potgieter, Malatje, Gaigher, & Venter, 2010), the term itself is ambiguous. The finding suggested the participants' inaccurate task understanding was influenced by their decision to draw knowledge and strategies from solving various discrete mathematics problems. Such phenomenon is commonly known as confidence bias, which is defined as "a systematic error of judgment made by individuals when they assess the correctness of their responses to questions relating to intellectual or perceptual problems" (Pallier et al., 2002, p.258). The participants' confidence bias was also worsened by the desire to exhibit their skills or abilities. As LStew explained, "I was trying to be smart and look/sound clever and focusing on what strengths I have that make me look smart (such as pattern recognition)."

Aside from having a confidence bias, Rusty, Anne, and LStew assumed that they could find a working pattern after conducting a few manual simulations of the provided suicidal method. They were so confident that they did not make any alternative plans. During the interview, Rusty said, "Well, if they *[educators]* are asking this question, there has got to be a systematic way to approach it; there has got to be some underlying pattern." Rusty's belief suggested the participants had positive assumptions about the academic environment. In their study of an authentic and impactful problem, Saulnier & Brisson (2018) reported a similar finding. In their study, students were asked to design a stove to be used in a four-day expedition. Some students did not take this task seriously, and most of them were shocked when they had to use their designed stove during the expedition. Plausibly, students are too used to working with inauthentic problems. McNeill, Douglas, Koro-Ljungberg, Therriault, & Krause's (2016) report confirmed this suspicion. They reported students believe "classroom problems are more abstract and less complex because of the need to simplify in order to provide a general understanding in school" (p.547).

4.2 Answering Research Question 2: How did their Initial Understanding Change during the Problem-Solving?

This subsection focuses on Rusty's problem-solving approach because he was the only participant who had a revised TI. Rusty's approach is simplified as a flowchart in Figure 2. Unlike a typical flowchart, the first and last activities were assumed as the first and last boxes. Thus, Rusty began solving the problem by simulating the provided suicidal method manually for inputs (i.e., number of people) of 5 to 17. He made a table to visualize each input and the associated output and tried to generate a working pattern. Unfortunately, it was not as straightforward as Rusty initially thought, and instead found out he skipped simulating the result for input 16. Rusty then fixed that error and continued determining a pattern. Again, he was unable to find any useful mathematical pattern. Rusty then reread the provided example and noticed a mismatch between that and his simulation steps. Rusty's verification confirmed nothing alarming, so he unsuccessfully continued trying to identify a pattern. Rusty then considered stopping looking for a pattern and implementing the provided method as it is. However, before doing so, he concluded there was no harm in rereading the problem description. The review supported Rusty's idea of implementing the method as described. Then, by using his C++ expertise, Rusty developed, verified, and submitted the pseudocode. Unfortunately, his submitted solution was still inaccurate because he did not print out the pseudocode's states. However, Rusty said during the interview that the description presented "a possible visualization of what it [the pseudocode] was doing," which suggested he would be able to get a complete task understanding and submit a correct solution given enough time (e.g., allowing him to work on this problem for several days).

While solving the Josephus problem, Rusty was observed engaging in 13 planning strategies, 29 enacting strategies, 131 monitoring activities, and 7 adjustment strategies. Further analysis revealed that out of his 138 observed monitoring and adjusting activities, 12 were related to TI. Theoretically, Rusty's revised task understandings were captured in these codes. However, since participants might sometimes forget to think aloud, some of their revised TI might be identified as planning or enacting strategies. Fortunately, the follow-up analysis confirmed all Rusty's observed engagements were aligned either with his initial or revised task TI. Throughout his endeavor, Rusty was observed revising his TI one time. In Figure 2, this event was presented as the fourth box. Thus, Rusty's approach suggested he rarely revised his task understanding during the problem-solving enterprise. Plausibly, this behavior was influenced by his familiarity with the problem.

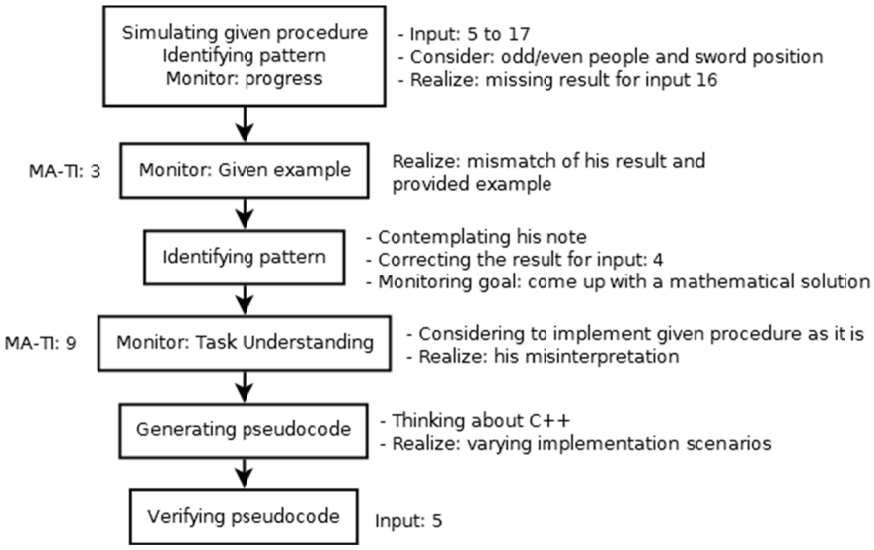

Figure 2. Rusty's approach to solving the Josephus problem (MA-TI: monitoring and adjusting of task interpretation)

4.3 Answering Research Question 3: What Were the Factors Influenced Those Changes?

This subsection focuses on Rusty's endeavor when solving the Josephus problem since he was the only participant to who had a revised TI during the data collection. He was observed changing his task understanding once and said:

"[Reading the problem description] You have to simulate each step... Oh my, gosh, I did not read that part thoroughly. [Reading the problem description] You have to simulate each step and then determine Josephus' position. Yes, so I was way overthinking it, [stressed his voice] way overthinking it."

After rereading the description, Rusty realized he did not read the description thoughtfully and generated an incorrect task understanding. The researchers explored this phenomenon during the interview by asking Rusty to explain the trigger that encouraged him to reread the problem description. Rusty replied:

"I kind of hit a cycle and I kept looping back to that [pattern], and I was like, okay, something is wrong, I am either not getting something, or there is something obvious that I am skipping over. … but it was not until I felt I had exhausted all my resources, best guesses, and ideas…"

Rusty's response suggested that the stagnancy of his problem-solving endeavor motivated him to reread the problem description. Interestingly, LStew's problem-solving enterprise was also stagnant, and although she engaged in TI-related monitoring activities, she did not change her understanding. Therefore, there could be another motivation that drove Rusty to think "I am either not getting something, or there is something obvious that I am skipping over" and then conquered his confidence bias.

Gigerenzer (1991) argues people develop normative claims, which are often oversimplified and thus misleading, that influence their problem-solving decisions. He further argues that helping people to be more competent in distinguishing "single-event confidences and relative frequencies" (p.89) may help in neutralizing their confidence bias. However, this argument was not applicable in this study because the participants' confidence bias was based on their discrete mathematics course experience, and in this course, most if not all of their assignments were asking them to develop mathematical formulas. Further, since many discrete mathematics concepts are commonly used in programming, establishing a connection with discrete mathematics intuitively or consciously was reasonable.

The researchers further explored this issue during the interview with Rusty, and he responded:

"That is something that I have done before with other programming assignments. I think I understood it [the assignments], and I did not read it carefully enough like I had the general idea [of the assignments], but I just jumped into it, and then I realized [my misinterpretation] halfway through [of solving the assignments]."

Rusty's response indicated he was aware of his tendency to miss essential small details when interpreting various programming assignments. This awareness was imbued in his conscience and influenced his problem-solving approach, which then helped him interpret the problem correctly. Plausibly, such belief was also a normative claim,

which unfortunately was not verified. Nevertheless, this finding suggested Rusty's understanding of himself, and his awareness and reflection of past failures, helped him conquered his confidence bias.

5. Conclusion, Implication, and Limitation

As a qualitative case study, this research had a limited number of participants. While having four participants is commonly considered unacceptable in quantitative research, it is not the case in the qualitative case study (Creswell, 2012). A qualitative case study is not designed to produce generalized findings but to collect as much diversity as possible (Creswell, 2012). In this study, the diversities are related to the senior CS students' TI while solving a variant of the Josephus problem. The descriptions of the participants and their approach are necessary to allow readers to better interpret and relate to the findings and apply it in their respective contexts.

In this study, two male and two female senior CS students at Utah State University were recruited. Three participants had prior exposure to a variant of the Josephus problem during their discrete mathematics course. Interestingly, all participants considered their experience in that course as relevant and drew strategies to solve the problem from it. This confidence bias negatively influenced their initial TI, in that it swayed the participants from having an accurate understanding of the task goal and from selecting an effective problem-solving approach. This confidence bias was also preventing Jake, Anne, and LStew from monitoring and adjusting their task understanding throughout the problem-solving enterprise. Contrarily, Rusty was able to defeat his confidence bias and acquire an accurate understanding of the task. The analysis suggested his awareness of himself and that he sometimes misinterprets the problems, played a crucial part in his success.

Considering that all participants had confidence bias suggested it was a common phenomenon in problem-solving. It is possible that one reason for the students' incompetence to address some of the assessment criteria is because they have a confidence bias. Reflecting on Rusty's experience, it might be beneficial to design an instruction that can incite students' confidence bias and then address the issue. The instruction will need to be memorable and encourage students to reflect on their experience. Exposing students to varieties of programming tasks might also useful in enhancing their competency to distinguish different task types. Naturally, follow-up studies are needed to learn more about the nature of confidence bias in programming.

Regarding the participants' diversity, there were no notable differences between male and female participants' initial TI and problem-solving approaches. However, the researchers found male participants spent twice as much time programming compared to the females. A dissimilarity was also found among the higher- and lower-performers' TI of the task requirements/constraints. Both higher-performer participants focused their interpretation on the implicit aspect of the task, while the lower-performers focused on the explicit aspect. It was also worth noting that even though all participants inaccurately identify the goal of the problem, not all of their follow-up TI were incorrect.

Since this report only focused on the Josephus problem, its findings only captured a small portion of a diverse students' SR while engaged in programming tasks. Although some findings might be transferable to other programming problem, replication studies are still needed to bring to light the other portion of students' self-regulations. Further, Maksel & Plucker (2014) and Benson & Borrego (2015) argue about the importance of replication studies, such as it can help to verify the accurateness and applicability of educational research findings and interventions in different settings. When conducting a replication study, it is important to focus on verbal responses instead of written to get more elaborate and detailed insight into the participants' thought processes.

Acknowledgments

This material is based, in part, upon work supported by the National Science Foundation under Grant No. 1148806. 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 National Science Foundation.

References

AAA Utah State University. (2016). *Summaries by Department*. Logan, UT, USA.

Abdillah, A., Nusantara, T., Subanj, S., Susanto, H., & Abadyo, A. (2016). The Students Decision Making in Solving Discount Problem. *International Education Studies*, *9*(7), 57. http://doi.org/10.5539/ies.v9n7p57

Adams, J. C. (2007). Alice, middle schoolers & the imaginary worlds camps. *ACM SIGCSE Bulletin*, *39*(1), 307. http://doi.org/10.1145/1227504.1227418

Alharbi, A., Henskens, F., & Hannaford, M. (2012). Student-Centered Learning Objects to Support the Self-Regulated Learning of Computer Science. *Creative Education*, *03*(06), 773-783. http://doi.org/10.4236/ce.2012.326116

Anderson, J. R., & Skwarecki, E. (1989). The automated tutoring of introductory computer programming. *Communications of the ACM*, *29*(9), 842-849.

Bainbridge, L., & Sanderson, P. (2005). Verbal Protocol Analysis. In J. R. Wilsom & N. Corlett (Eds.), *Evaluation of Human Work, 3rd Edition* (3rd ed., p. 1048). CRC Press. Retrieved from https://books.google.com/books?id=dSmKYLp82b4C&pgis=1

Ball, L. J., Ormerod, T. C., & Morley, N. J. (2004). Spontaneous analogising in engineering design: a comparative analysis of experts and novices. *Design Studies*, *25*(5), 495-508. http://doi.org/10.1016/j.destud.2004.05.004

Beaubouef, T., & Mason, J. (2005). Why the high attrition rate for computer science students. *ACM SIGCSE Bulletin*, *37*(2), 103. http://doi.org/10.1145/1083431.1083474

Ben-Ari, M. (1998). Constructivism in Computer Science Education. *ACM SIGCSE Bulletin*, *30*(1), 257-261. http://doi.org/10.1145/274790.274308

Benson, L., & Borrego, M. (2015). The Role of Replication in Engineering Education Research. *Journal of Engineering Education*, *104*(4), 388-392. http://doi.org/10.1002/jee.20082

Bergin, S., Reilly, R., & Traynor, D. (2005). Examining the role of self-regulated learning on introductory programming performance. In *First International Workshop on Computing Education Research* (pp. 81-86). New York, New York, USA: ACM Press. http://doi.org/10.1145/1089786.1089794

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *... of the 2012 Annual Meeting of ...*. Retrieved from http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Briggs, T. (2005). Techniques for active learning in CS courses. *Journal of Computing Sciences in Colleges*, *21*(2), 156-165. Retrieved from http://dl.acm.org/citation.cfm?id=1089053.1089075

Bui, Q. (2015). Will Your Job Be Done By A Machine? Retrieved May 25, 2015, from http://www.npr.org/sections/money/2015/05/21/408234543/will-your-job-be-done-by-a-machine

Butler, D. L. (1995). Promoting Strategic Learning by Postsecondary Students with Learning Disabilities. *Journal of Learning Disabilities*, *28*(3), 170-190. http://doi.org/10.1177/002221949502800306

Butler, D. L. (1998). Metacognition and Learning Disabilities. In B. Y. L. Wong (Ed.), *Learning About Learning Disabilities* (2nd ed., pp. 277-307). Toronto: Academic Press.

Butler, D. L., & Cartier, S. C. (2004). Learning in varying activities: An explanatory framework and a new evaluation tool founded on a model of self-regulated learning. In *Annual Conference of the Canadian Society for The Study of Education*. Toronto, ON. Retrieved from http://perso.crifpe.ca/~scartier/spip/IMG/pdf/Butler_and_Cartier_2004_.pdf

Butler, D. L., & Cartier, S. C. (2005). Multiple Complementary Methods for Understanding Self-Regulated Learning as Situated in Context. In *American Educational Research Association, Annual Meeting* (pp. 11-15).

Butler, D. L., & Cartier, S. C. (2018). Case Studies as a Methodological Framework for Studying and Assessing Self-Regulated Learning. In D. H. Schunk & J. Greene (Eds.), *Handbook of Self-Regulation of Learning and Performance* (2nd ed., pp. 352-369). New York, New York, USA: Routledge.

Butler, D. L., Schnellert, L., & MacNeil, K. (2015). Collaborative inquiry and distributed agency in educational change: A case study of a multi-level community of inquiry. *Journal of Educational Change*, *16*(1), 1-26. http://doi.org/10.1007/s10833-014-9227-z

Butler, D. L., Schnellert, L., & Perry, N. E. (2017). *Developing Self-Regulating Learners*. (C. O'Donnell, Ed.). Toronto, ON, Canada: Pearson Education Inc.

Butler, D. L., & Winne, P. H. (1995). Feedback and Self-Regulated Learning: A Theoretical Synthesis. *Review of Educational Research*, *65*(3), 245-281. http://doi.org/10.3102/00346543065003245

Cartier, S. C., & Butler, D. L. (2004). Elaboration and validation of questionnaires and plan for analysis. In *Annual Conference of the Canadian Society for The Study of Education*. Toronto, ON.

Chi, M. T. H., De Leeuw, N., Chiu, M.-H., & Lavancher, C. (1994). Eliciting Self-Explanations Improves Understanding. *Cognitive Science*, *18*(3), 439-477. http://doi.org/10.1207/s15516709cog1803_3

Coutinho, S. A. (2007). The relationship between goals, metacognition, and academic success. *Educate*, *7*(1),

39-47. Retrieved from http://www.educatejournal.org/index.php/educate/issue/view/23

Creswell, J. W. (2012). *Qualitative Inquiry and Research Design: Choosing Among Five Approaches* (3rd ed.). SAGE Publications.

Denning, P. J. (2003). Computer Science. In A. Ralston, E. D. Reilly, & D. Hemmendinger (Eds.), *Encyclopedia of Computer Science* (4th ed., pp. 405-419). Chichester, UK: John Wiley and Sons Ltd. Retrieved from http://dl.acm.org/citation.cfm?id=1074266

Dent, A. L., & Koenka, A. C. (2016). The Relation Between Self-Regulated Learning and Academic Achievement Across Childhood and Adolescence: A Meta-Analysis. *Educational Psychology Review*, *28*(3), 425-474. http://doi.org/10.1007/s10648-015-9320-8

Dinsmore, D. L., Alexander, P. A., & Loughlin, S. M. (2008). Focusing the conceptual lens on metacognition, self-regulation, and self-regulated learning. *Educational Psychology Review*, *20*(4), 391-409. http://doi.org/10.1007/s10648-008-9083-6

Falkner, K., Szabo, C., Michell, D., Szorenyi, A., & Thyer, S. (2015). Gender Gap in Academia: Perceptions of Female Computer Science Academics. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE '15* (pp. 111-116). New York, New York, USA: ACM Press. http://doi.org/10.1145/2729094.2742595

Falkner, K., Vivian, R., & Falkner, N. J. G. (2014). Identifying computer science self-regulated learning strategies. In *Proceedings of the 2014 conference on Innovation & technology in computer science education - ITiCSE '14* (pp. 291-296). New York, New York, USA: ACM Press. http://doi.org/10.1145/2591708.2591715

Febrian, A., Lawanto, O., & Cromwell, M. (2015). Advancing Research on Engineering Design using e-Journal. In *Frontiers in Education Conference (FIE), 2015. 32614 2015. IEEE* (pp. 1-5). El Paso, TX: IEEE. http://doi.org/10.1109/FIE.2015.7344191

Felder, R. M., & Soloman, B. A. (n.d.). Learning Styles and Strategies. Retrieved April 25, 2017, from http://www4.ncsu.edu/unity/lockers/users/f/felder/public//ILSdir/styles.htm

Gal-Ezer, J., & Harel, D. (1998). What (else) should CS educators know? *Communications of the ACM*, *41*(9), 77-84. http://doi.org/10.1145/285070.285085

Gigerenzer, G. (1991). How to Make Cognitive Illusions Disappear: Beyond "Heuristics and Biases." *European Review of Social Psychology*, *2*(1), 83-115. http://doi.org/10.1080/14792779143000033

Gonzalez, G. (2006). A systematic approach to active and cooperative learning in CS1 and its effects on CS2. In *ACM SIGCSE Bulletin* (Vol. 38, p. 133). New York, New York, USA: ACM Press. http://doi.org/10.1145/1124706.1121386

Graham, S., & Latulipe, C. (2003). CS girls rock: sparking interest in computer science and debunking the stereotypes. *ACM SIGCSE Bulletin*, *35*(1), 322. http://doi.org/10.1145/792548.611998

Gronlund, N. E., Gronlund, E. N., & Waugh, C. K. (2013). *Assessment of Student Achievement*. *Assessment of Student Achievement* (10th ed.). Pearson.

Guzdial, M., Johnson, R., Wampler, K., Kussmaul, C., Swanson, J., Humenn, P., & Lewchuk, M. (2015). What's the best way to teach computer science to beginners? *Communications of the ACM*, *58*(2), 12-13. http://doi.org/10.1145/2714488

Hadwin, A. (2006). Do your students really understand your assignments? *LTC Currents: Optimizing Learning Environments*, *11*(3), 8-9.

Hadwin, A. F., Jarvela, S., & Miller, M. (2011). Self-regulated, co-regulated, and socially shared regulation of learning. In B. J. Zimmerman & D. H. Schunk (Eds.), *Handbook of Self-Regulation of Learning and Performance* (pp. 65-84). New York, New York, USA: Routledge.

Hadwin, A. F., Oshige, M., Miller, M., & Wild, P. (2009). Examining Student and Instructor Task Perceptions in a Complex Engineering Design Task. In *The Sixth International Conference on Innovation and Practices in Engineering Design and Engineering Education*. Hamilton, ON, Canada: McMaster University.

Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). A Multidisciplinary Approach Towards Computational Thinking for Science Majors. *ACM SIGCSE Bulletin*, *41*(1), 183. http://doi.org/10.1145/1539024.1508931

Havenga, M. (2015). The Role of Metacognitive Skills in Solving Object-Oriented Programming Problems: a Case Study. *TD: The Journal for Transdisciplinary Research in Southern Africa*, *11*(1), 133-147. Retrieved from https://journals.co.za/content/transd/11/1/EJC175923

Henderson, P. B. (2009). Ubiquitous computational thinking. *Computer*, *42*(10), 100-102. http://doi.org/10.1109/MC.2009.334

Howles, T. (2007). Preliminary Results of a Longitudinal Study of Computer Science Student Trends, Behaviors and Preferences. *Journal of Computing Sciences in Colleges*, *22*(6), 18-27. Retrieved from http://dl.acm.org/citation.cfm?id=1231097

Irani, L. (2004). Understanding gender and confidence in CS course culture. *ACM SIGCSE Bulletin*, *36*(1), 195. http://doi.org/10.1145/1028174.971371

Isomöttönen, V., & Tirronen, V. (2013). Teaching programming by emphasizing self-direction. *ACM Transactions on Computing Education*, *13*(2), 1-21. http://doi.org/10.1145/2483710.2483711

Johnson, S. D. (2008). Cognitive Analysis of Expert and Novice Troubleshooting Performance. *Performance Improvement Quarterly*, *1*(3), 38-54. http://doi.org/10.1111/j.1937-8327.1988.tb00021.x

Jonassen, D. H. (2000). Toward a design theory of problem solving. *Educational Technology Research and Development*, *48*(4), 63-85. http://doi.org/10.1007/BF02300500

Jonassen, D. H. (2004). *Learning to Solve Problems: An Instructional Design Guide*. (M. Davis, Ed.). John Wiley & Sons.

Jonassen, D. H. (2010). *Learning to solve problems: A handbook for designing problem-solving learning environments*. *Learning to Solve Problems: A Handbook for Designing Problem-Solving Learning Environments*. Routledge. http://doi.org/10.4324/9780203847527

Kahneman, D. (2003). A Perspective on Judgment and Choice: Mapping Bounded Rationality. *American Psychologist*, *58*(9), 697-720. http://doi.org/10.1037/0003-066X.58.9.697

Kori, K., Pedaste, M., Tonisson, E., Palts, T., Altin, H., Rantsus, R., … Ruutmann, T. (2015). First-year dropout in ICT studies. In *2015 IEEE Global Engineering Education Conference (EDUCON)* (pp. 437-445). IEEE. http://doi.org/10.1109/EDUCON.2015.7096008

Krauss, J. (2008). *Computer Science-in-a-Box: Unplug Your Curriculum*. (D. Burkhart & C. Stephenson, Eds.). Boulder, CO: The National Center for Women & Information Technology. Retrieved from https://www.ncwit.org/resources/computer-science-box-unplug-your-curriculum

Kumar, V., Winne, P., Hadwin, A., Nesbit, J., Jamieson-Noel, D., Calvert, T., & Samin, B. (2005). Effects of self-regulated learning in programming. In *Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05)* (pp. 383-387). IEEE. http://doi.org/10.1109/ICALT.2005.131

Lacey, T. A., & Wright, B. (2009). *Monthly Labor Review: Occupational Employment Projections to 2018*.

Lapadat, J. C., & Lindsay, A. C. (1999). Transcription in Research and Practice: From Standardization of Technique to Interpretive Positionings. *Qualitative Inquiry*, *5*(1), 64-86. http://doi.org/10.1177/107780049900500104

Lawanto, O. (2010). Students' metacognition during an engineering design project. *Performance Improvement Quarterly*, *24*(2), 115-134.

Lawanto, O., Butler, D., Cartier, S. C., Santoso, H. B., Goodridge, W., Lawanto, K. N., & Clark, D. (2013). Pattern of Task Interpretation and Self-Regulated Learning Strategies of High School Students and College Freshmen during an Engineering Design Project. *Journal of STEM Education: Innovations and Research*, *14*(4), 34-46.

Lawanto, O., Butler, D., Cartier, S., Santoso, H. B., & Goodridge, W. (2013). Task Interpretation, Cognitive, and Metacognitive Strategies of Higher and Lower Performers in an Engineering Design Project: An Exploratory Study of College Freshmen. *International Journal of Engineering Education*, *29*(2), 459-475.

Lawanto, O., Cromwell, M., & Febrian, A. (2016). Students' Self-Regulation in Managing Their Capstone Senior Design Projects. In *123rd ASEE Annual Conference and Exposition*. New Orleans, LA, USA.

Lawanto, O., & Johnson, S. (2009). Student's cognitive self-appraisal, self-management, and the level of difficulty of an engineering design project: are they related? In *American Society for Engineering Education Annual Conference*. Austin, TX.

Lee, K. D. (2014). *Foundations of Programming Languages*. Cham: Springer International Publishing. http://doi.org/10.1007/978-3-319-13314-0

Lewis, C. M., Anderson, R. E., & Yasuhara, K. (2016). "I Don't Code All Day": Fitting in Computer Science When the Stereotypes Don't Fit. In *Proceedings of the 2016 ACM Conference on International Computing Education Research - ICER '16* (pp. 23-32). New York, New York, USA: ACM Press. http://doi.org/10.1145/2960310.2960332

Lischner, R. (2001). Explorations: Structured Labs for First-Time Programmers. *ACM SIGCSE Bulletin*, *33*(1), 154-158. http://doi.org/10.1145/366413.364571

Lonchamp, J. (2010). Customizable Computer-based Interaction Analysis for Coaching and SelfRegulation in Synchronous CSCL Systems. *Educational Technology & Society*, *13*(2), 193-205.

Lu, B., Conley, M., Klein, A., & Drive, B. J. (2014). Teaching True Computer Science Principles To General Students. *Journal of Computing Sciences in Colleges*, *29*(5), 233-239. Retrieved from http://dl.acm.org/citation.cfm?id=2600623.2600668

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51-61. http://doi.org/10.1016/j.chb.2014.09.012

Madigan, E. M., Goodfellow, M., & Stone, J. A. (2007). Gender, perceptions, and reality: technological literacy among first-year students. *ACM SIGCSE Bulletin*, *39*(1), 410. http://doi.org/10.1145/1227504.1227453

Makel, M. C., & Plucker, J. A. (2014). Facts Are More Important Than Novelty. *Educational Researcher*, *43*(6), 304-316. http://doi.org/10.3102/0013189X14545513

Mata, A., Ferreira, M. B., & Sherman, S. J. (2013). The metacognitive advantage of deliberative thinkers: A dual-process perspective on overconfidence. *Journal of Personality and Social Psychology*, *105*(3), 353-373. http://doi.org/10.1037/a0033640

McNeill, N. J., Douglas, E. P., Koro-Ljungberg, M., Therriault, D. J., & Krause, I. (2016). Undergraduate Students' Beliefs about Engineering Problem Solving. *Journal of Engineering Education*, *105*(4), 560-584. http://doi.org/10.1002/jee.20150

Outlay, C. N., Platt, A. J., & Conroy, K. (2017). Getting IT Together: A Longitudinal Look at Linking Girls' Interest in IT Careers to Lessons Taught in Middle School Camps. *ACM Transactions on Computing Education*, *17*(4), 1-17. http://doi.org/10.1145/3068838

Pallier, G., Wilkinson, R., Danthiir, V., Kleitman, S., Knezevic, G., Stankov, L., & Roberts, R. D. (2002). The Role of Individual Differences in the Accuracy of Confidence Judgments. *The Journal of General Psychology*, *129*(3), 257-299. http://doi.org/10.1080/00221300209602099

Peng, J., Wang, M., & Sampson, D. (2017). Visualizing the Complex Process for Deep Learning with an Authentic Programming Project. *Educational Technology & Society*, *20*(4), 275-287.

Pintrich, P. R. (2002). The role of metacognitive knowledge in learning, teaching, and assessing. *Theory into Practice*, *41*(4), 231-236.

Pivkina, I., Pontelli, E., Jensen, R., & Haebe, J. (2009). Young Women in Computing: Lessons Learned from an Educational & Outreach Program. In *Proceedings of the 40th ACM technical symposium on Computer science education - SIGCSE '09* (Vol. 41, p. 509). New York, New York, USA: ACM Press. http://doi.org/10.1145/1508865.1509042

Potgieter, M., Malatje, E., Gaigher, E., & Venter, E. (2010). Confidence versus Performance as an Indicator of the Presence of Alternative Conceptions and Inadequate Problem‐Solving Skills in Mechanics. *International Journal of Science Education*, *32*(11), 1407-1429. http://doi.org/10.1080/09500690903100265

Renumol, V. G., Janakiram, D., & Jayaprakash, S. (2010). Identification of Cognitive Processes of Effective and Ineffective Students During Computer Programming. *ACM Transactions on Computing Education*, *10*(3), 1-21. http://doi.org/10.1145/1821996.1821998

Resnick, M., Silverman, B., Kafai, Y., Maloney, J., Monroy-Hernández, A., Rusk, N., … Silver, J. (2009). Scratch: Programming for All. *Communications of the ACM*, *52*(11), 60. http://doi.org/10.1145/1592761.1592779

Rivera-Reyes, P. (2015). Students' Task Interpretation and Conceptual Understanding in Electronics Laboratory Work (Doctoral Dissertation). Logan, UT: Utah State University.

Rivera-Reyes, P., Lawanto, O., & Pate, M. L. (2016). Understanding Student Coregulation in Task Interpretation during Electronics Laboratory Activities. *International Education Studies*, *9*(7), 1. http://doi.org/10.5539/ies.v9n7p1

Rivera-Reyes, P., Lawanto, O., & Pate, M. L. (2017). Students' Task Interpretation and Conceptual Understanding in an Electronics Laboratory. *IEEE Transactions on Education*, *60*(4), 265-272. http://doi.org/10.1109/TE.2017.2689723

Rum, S. N. M., & Ismail, M. A. (2017). Metacognitive Support Accelerates Computer Assisted Learning for Novice Programmers. *Educational Technology & Society*, *20*(3), 170-181.

Ruthmann, A., Heines, J. M., Greher, G. R., Laidler, P., & Saulters, C. (2010). Teaching computational thinking through musical live coding in scratch. In *Proceedings of the 41st ACM technical symposium on Computer science education - SIGCSE '10* (p. 351). New York, New York, USA: ACM Press. http://doi.org/10.1145/1734263.1734384

Santoso, H. B. (2013). Computer Self-Efficacy, Cognitive Actions, and Metacognitive Strategies of High School Students While Engaged in Interactive Learning Modules (Doctoral Dissertation). Logan, UT: Utah State University. Retrieved from http://discover.lib.usu.edu/iii/encore/record/C__Rb3243218__Sharry budi santoso__Orightresult__X4?lang=eng&suite=cobalt

Saulnier, C. R., & Brisson, J. G. (2018). Design for Use: A Case Study of an Authentically Impactful Design Experience. *International Journal of Engineering Education*, *34*(2B), 769-779.

Schoenfeld, A. H. (1983). Episodes and Executive Decisions in Mathematical Problem Solving. In R. Lesh & M. Landau (Eds.), *Acquisition of Mathematics Concepts and Processes* (pp. 345-395). New York, New York, USA.

Tigerfish. (n.d.). Transcription Style Guide. San Francisco, USA: Tigerfish. Retrieved from http://www.tigerfish.com/Transcription Style Guide Rev. 6.10.pdf

Viera, A. J., & Garrett, J. M. (2005). Understanding interobserver agreement: the kappa statistic. *Family Medicine*, *37*(5), 360-363. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/15883903

Wang, J., Hejazi Moghadam, S., & Tiffany-Morales, J. (2017). Social Perceptions in Computer Science and Implications for Diverse Students. In *Proceedings of the 2017 ACM Conference on International Computing Education Research - ICER '17* (pp. 47-55). Tacoma, Washington, USA: ACM Press. http://doi.org/10.1145/3105726.3106175

Wang, X.-M., Hwang, G.-J., Liang, Z.-Y., & Wang, H.-Y. (2017). Enhancing Students' Computer Programming Performances, Critical Thinking Awareness and Attitudes towards Programming: An Online PeerAssessment Attempt. *Educational Technology & Society*, *20*(4), 58-68.

Whittington, K. J. (2004). Infusing Active Learning Into Introductory Programming Courses. *Journal of Computing Sciences in Colleges*, *19*(5), 249-259. Retrieved from http://dl.acm.org/citation.cfm?id=1060081.1060111

Williams, L., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2010). In Support of Pair Programming in the Introductory Computer Science Course. *Computer Science Education*, *12*(3), 197-212. http://doi.org/10.1076/csed.12.3.197.8618

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*, *49*(3), 33. http://doi.org/10.1145/1118178.1118215

Wing, J. M. (2008). Computational Thinking and Thinking About Computing. *Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences*, *366*(1881), 3717-25. http://doi.org/10.1098/rsta.2008.0118

Zimmerman, B. J. (2008). Investigating Self-Regulation and Motivation: Historical Background, Methodological Developments, and Future Prospects. *American Educational Research Journal*, *45*(1), 166-183. http://doi.org/10.3102/0002831207312909

Zimmerman, B. J., Heart, N., & Mellins, R. B. (1989). A Social Cognitive View of Self-Regulated Academic Learning. *Journal of Educational Psychology*, *81*(3), 329-339. http://doi.org/10.1037//0022-0663.81.3.329