Global Journals LaTeX JournalKaleidoscope™

Artificial Intelligence formulated this projection for compatibility purposes from the original article published at Global Journals. However, this technology is currently in beta. *Therefore, kindly ignore odd layouts, missed formulae, text, tables, or figures.*

Students' Understanding of an Object-Oriented Design Task -A Case Study

Oenardi Lawanto

Received: 15 December 2019 Accepted: 4 January 2020 Published: 15 January 2020

Abstract

Students must understand a problem accurately to solve it correctly. Unfortunately, numerous studies reported that students only have a partial understanding of the information presented in the problem description, including in computer science. This study assesses students' task and revised-task interpretations when working on an objectoriented design problem. Multiple qualitative case study research was used in this study. Two male1 I. Introduction and two female senior computer science students at Utah State University, USA, volunteered as participants. They were asked to solve five programming problems while thinking aloud, complete surveys, and answer several interview questions. The study found that the participants were able to identify most of the essential information after the initial reading of the problem description. They strategically ignore detailed information that may affect their design decisions and update it throughout their problem-solving enterprise.

***Index terms*—** cognition, problem-solving, programming, self-regulation, self-regulated learning, task interpretation, task revision.

1 Introduction

and two female senior computer science students at Utah State University, USA, volunteered as participants. They were asked to solve five programming problems while thinking aloud, complete surveys, and answer several interview questions. The study found that the participants were able to identify most of the essential information after the initial reading of the problem description. They strategically ignore detailed information that may affect their design decisions and update it throughout their problem-solving enterprise.

Index terms: cognition, problem-solving, programming, self-regulation, self-regulated learning, task interpretation, task revision. t was a typical day in a programming lab session; students were working on their task under the observation of several teaching assistants. Several students concentrated on solving the lab problem, some were discussing the best approach to solve it, and some others were waiting for the answer from their peers. Interestingly, some students did not even bother to open and read the lab instruction, regardless of suggestion and encouragement from the assistants. While the motivation for their persistence may vary, reading and rereading a problem is a crucial step to understand and solve it [1]- [5].

To accurately understanding a problem is not an easy task. Several studies reported that students are rarely able to interpret a problem correctly [2], [3], [6]- [8].Some studies also reported that students' submitted solutions reveal their incomplete understanding of the 1 This paragraph of the first footnote will contain the date on which you submitted your paper for review. It will also contain support information, including sponsor and financial support acknowledgment. For exam ple, "This work was supported in part by the U.S. Department of Commerce under Grant BS123456." 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.

given tasks [2], [8], [9].Although limited in number, similar phenomena also have been reported in the discipline of computer science (CS).Some CS students were reported incapable of accurately inferring the expected program's behaviors based on a given design brief [10]. Other study reported that CS students tend to ignore some assessment criteria while working on their tasks, which then negatively impact their grades [11].

In this study, we aim to describe the approaches used by senior CS students in understanding an object-oriented (OO) design problem; i.e., their initial task interpretation and the changes. Selfregulated learning (SRL) framework is used to distinguish their cognitive and metacognitive activities during the problem-solving endeavor. The description and analysis results may help instructors to understand better, and encourage students to enhance their strategies in comprehending a design problem. The description may also help students to be more aware of their self-regulation so that they can improve it.

2 II.

3 Research Questions

As mentioned earlier, this study aims to describe senior CS students' approaches to understanding an OO design problem. In more specific, this study intends to assess (1) students' initial explicit and implicit task understanding, (2) how their initial understanding changes during the problem-solving activity, and (3) identify factors that influence those changes.

4 III.

5 Relevant Literature

Since this study uses SRL as a framework in analyzing the data, the literature will discuss task understanding (or task interpretation) within the SRL. Additionally, this section also discusses known literature on self-regulation in CS to help readers familiar with existing research in that area.

6 a) Task Interpretation in Self-Regulated Learning

Students deliberately self-regulate when working on a task [12], [13]. Such activity involves the interplay of interpreting a given task, developing a plan, and executing, monitoring, and adjusting the plan to complete the task [4], [5], [13]- [16]. Fig. **??** and Table I presents the relationship and definition of each SRL activity, respectively. It is clear from Fig. **??** that task interpretation, which refers to understanding the task and associated process to complete it [17], is the starting point of any SRL activities. Thus, misinterpreting a task may negatively affect follow-up planning, enacting, monitoring, and adjusting activities [18].

Fig. **??**: Categories of various self-regulation activities. When interpreting a task, one must consider the explicit and implicit aspects of it. Explicit task interpretation refers to students' understanding of the information presented in the problem description [8], such as written goals, requirements, and constraints. Implicit task interpretation refers to extrapolated information base on the given description [8], for example, relevant concepts and experience to solve the problem. These definitions imply that explicit and implicit task interpretation is distinguishable based on the manner of that specific understanding being acquired (i.e., by identifying or extrapolating).

Unfortunately, interpreting a task is not easy. Two studies reported that students could only correctly identify 63% -77% of valuable information presented in Thermodynamics course problems [3], [7]. The accuracy of implicit task understanding is even more unsatisfactory, such that they could only extrapolate 37% -49% of the essential information [3], [7]. Similar findings have been reported in engineering design [8] and electronics lab [2]. Consequently, this misinterpretation impedes their problem-solving performance [19]- [21].

Fortunately, several studies [2], [3], [8], [19], [21], [22] suggested that students enhance their task understanding throughout their problem-solving enterprise. Theoretically, these refinements occur due to continuous monitoring and adjusting activities [23]. Thus, insufficient and inefficient monitoring and adjustment activities may lead to a meager solution. Since SRL is contextual, having sufficient relevant domain knowledge is necessary for efficient monitoring and adjustment activities [4], [8], [19], [24]. Moreover, one has to be willing to adopt new interpretations or strategies when it is necessary to do so.

7 b) Self-Regulation in Computer Programming

Although still limited in number, SRL research in CS is not new. Some scholars believe that it may ease the curve of learning programming and increase student's retention rate [25]. In this section, the reported cognitive and meta cognitive characteristics of CS students found in the literature are discussed.

Most CS students prefer to learn new materials sequentially through visual representation, and then reflect on their progress [26]. Most of them are comfortable and competent in dealing with detailed information [26],which strengthens their ability to solve complex problems (e.g., developing software systems). Their reflective nature allows them to be appreciative of each task, which, in turn, influences them to be more self-regulated and deliver better outputs [27].

A study reported that students use numerous SRL strategies instinctively when trying to understand a task, design a solution, and debug a program [28]. Engaging in self-regulation activities may improve their performance [27], [29].One study reported that students are sometimes unable to accurately address all the requirements and constraints of a problem [11], which suggested that instinctive self-regulation may not be sufficient in the long run. Students need to be more conscious of using it. Two studies suggested that deepening students' familiarity with various programming concepts and principles (i.e., contexts or knowledge) may increase their SRL quality [11], [19].

Related to object-oriented (OO) design, a study reported that students are using typically suggested strategies in interpreting an OO design problem, which is by identifying the nouns and verbs found in the task description [9]. Although this report seems expected, this finding is important because it describes students ' approaches in design, not just a belief. Based on their understanding, students then decompose the problem and design the solution. Interestingly, students consider problem decomposition as a skill that hard to master [11]. Students tend to have incomplete and incorrect knowledge about OO design [9], which, plausibly, impair their decomposition skills. While some students may be aware of their weaknesses and strive to address it, others chose to ignore it. The last group of students tends to feel discouraged when facing a challenge [9] and, thus, have a negative learning experience.

8 IV.

9 Research Design

In this section, the research design and its justification were explicated, which include data Selecting strategies to complete the task [5] Enacting Strategies Students' cognitive activities employed while completing the task [54] collection and analysis methods, and the design problem.

10 a) Data Collection Method

Multiple, in-depth qualitative data were collected from the participants, which aligned with best practices of qualitative study [30] and conducting SRL research [5], [31], [32]. Multiple data points allowed the researchers to appraise the perception and activities of the participants accurately.

Five programming problems, five problemspace maps, initial task interpretation survey, and interview question templates, were developed, pilot tested, and used. The programming problems consisted of two practice, one OO, one break, and one algorithm tasks. All except one were related to imperative programming paradigms. The problem-space maps described all correct and possible explicit and implicit task interpretation of each problem. This technique was adopted from expert-novice research about trouble shooting [33]. The initial task interpretation survey was used to assess the participants' initial understanding of the task. Table II presents the survey questions and the associated aspect of task interpretation. The interview question templates were used to formulating confirmatory questions based on the researchers' observation.

During the data collection, the participants followed a specific protocol when solving each programming problem, and were audio-and videorecorded. The participants observed the following protocol in sequence: (a) reading the problem description aloud, (b) completing initial task interpretation survey while thinking aloud, (c) continue solving the problem while thinking aloud, and (d) answering the interview questions. The programming problems were given in the order written in the previous paragraph. No time limit was set for each problem. The practice problems were used to help the participants familiar with the data collection protocol and address any thinking aloud issues, if any.

To accurately capture the initial task interpretation, the participants were prohibited from rereading the problem description when completing the survey (i.e., step (a)). The problem-space maps were used to track the participants' thought processes when solving the problem (i.e., step (c)). The interview was semi-structured to ensure its alignment with the research goal yet still providing flexibility in pursuing particular points of interest that emerged during the problem-solving process.

11 b) Object-Oriented Design Problem

The OO problem is about designing a digital version of a classic board game, which commonly known as the Monopoly. Unlike the original, this game would be set in Middle-Ages. Given a set of requirements and constraints (see Table III), the participants should design a game base so that the rest of the team members could move forward smoothly. They are expected to deliver a class diagram. Also, they are allowed to ignore animation and play-testing parts and add their creativity beyond the given requirements and constraints.

The participants are expected to declare and manage at least one function, five issues, and 4 to 41 variables when solving this problem. It also contains some missing or unspecified information (i.e., implicit task interpretation) and has multiple solutions; all are typical characteristics of a design problem [34]- [36]. Consequently, the participants are not expected to comprehend the problem in one read. Based on the revised Bloom's Taxonomy [37], this problem belonged to the creation category, where the participants were expected to make a product for a specific purpose.

12 c) Data Analysis Method

Recorded video/audio files, initial task interpretation survey responses, design solutions, design notes (if any), observed thought processes (i.e., problem-space maps), and interview responses were collected from each participant. All recorded video/audio files were transcribed using the verbatim technique, such that the transcriptions recorded all articulated words and shutters [38]. Three additional notations were introduced in the transcriptions to clarify relevant contexts, including square bracket ("[]"), dash ("-"), and capitalizing the first letter for describing the participants' actions, correcting statements, and clarifying programming concepts, respectively. For example, "Since not having a particular idea on The qualitative coding process consisted of two phases. In the first phase, both experts individually coded the transcriptions based on the definition provided in Table I. After they finished, the coding results were then combined. Some disagreements were expected since the experts worked independently. In the second phase, the experts met face-to-face to discuss and resolve all coding disagreements. All collected data were used to ensure correct interpretations of the participants' statements. Through this process, the experts were able to reach a perfect agreement, with a Kappa score of 1.00 for each transcription, and produced 875 codes.

To answer the first research question, the initial task interpretation survey responses and the associated recorded video/audio files were used. These data sources were also triangulated against recorded problem-solving approaches and interview responses. This step was necessary since the participants might forget reporting all relevant thought processes when answering the survey.

To answer the second research question, the answer to the first research question and the codedtranscriptions were used. All problem-solving activities that could not be associated with the initial task interpretation were categorized as adjustment of participants' task understanding. These adjusted interpretations were then triangulated against recorded interview responses.

To answer the third research question, the list of task interpretation adjustments, coded-transcriptions, and interview responses were used. All statements in the coded-transcription that were associated with the changes were marked. The factors that influence the marked changes were then identified and triangulated against the interview responses.

V.

13 The Participants

After the transcribing process completed, the OO-related transcriptions were qualitatively coded by two experts, which were an information technologist and one of the researchers. All experts had experience in developing OO applications. The expert-researcher also had a bachelor's and master's degrees in CS.

14 Table III: Object-Oriented Problem Requirements and Constraints

No.

Requirements and Constraints 1

The game is meant to be played by either two, three, or four players. 2

Each player chooses to play as any one of the following characters: King, Warrior, Merchant, or Thief. Each character has unique special abilities and starts with different items and different amounts of money. 3

The game board will consist of 30 spaces where players can land, arranged in a circle. On some spaces, there are buildings that can be bought and sold. On other spaces, there are shops where players can buy items. In addition, some spaces have specific instructions that players must follow when they land there. 4

In the original board game, movement is determined by rolling dice, so you must develop an equivalent virtual method of determining the number of spaces each player moves on his or her turn. 5

On their turn, each player must move, and they can choose to do any of the following: buy the building on the space they are on, sell any building they own, spend money to improve buildings they own, or use one of their character's special abilities. 6

Items give special benefits to the player. Items include the following: Sword, Potion, Horse, or others. The effects of the item will be different for each character type. 7

There are three different kinds of buildings: Castle, F d I Th b ildi h diff fourth year, USU CS students have typically completed the introduction to programming, algorithm and data structure, software engineering, event-driven programming, and internship courses. At the end of the research, each participant received a personalized report of his or her task interpretation strategies and suggestions for improvement and a $40 gift card. Participants responded positively towards the reports and suggestions.

All participants were Caucasians with GPAs of 3.10 to 3.96 on a 4-point scale. Sorted based on their GPAs, they were Jake, Anne, LStew, and Rusty. The male participants also familiar with logic programming and had spent approximately 4980 hours developing their programming skills.

The female participants had spent about 2050 hours of programming. Similar to most female CS students [39]- [44], they had struggled with CS stereotypes, where CS students are viewed as overtly "focused on CS, asocial, competitive, and male" (p.30) [40]. They also suffered from comparing themselves against their peers. L Stew said, "I have to ignore my colleagues and classmates programming 'successes' as that comparison game tends to reduce my selfesteem a lot and negatively impact my problem-solving and programming capabilities." She also said, "I nearly failed a class because I did not believe I was capable of succeeding in it." Fortunately, both participants were able to overcome that challenge and were almost finished with the degree requirements.

15 VI.

16 Findings

All participants started with incomplete task understanding, which was expected, as explained earlier. Fortunately, all participants were also aware of it and tried to update their task understanding. Unfortunately, although their final task interpretation was better compared to the initial, it was still incomplete. There are two possible reasons for this result. First, the participants were overwhelmed with the detail of their design. Second, the participants were drawing knowledge from irrelevant experience. Rusty, for example, was using the entity-relationship instead of the class diagram.

In this section, participants' initial and revisedtask interpretation, and factors that influenced the changes were discussed.

17 a) Initial Explicit and Implicit Task Interpretation

Five questions were asked to assess the initial understanding of the participants (see Table II). All participants were able to determine the problem goal correctly. Anne, for example, defined the goal as "develop[ing a] class diagram from given constraints." L Stew and Rusty also included design best practices and their interest in the problem goal. Rusty, for example, said the problem goal was "create[ing] a logic layer inside of our program that can function completely without interaction from the graphical user interface or user."Rusty knew that the decoupling of logic and user interface is part of software design best practices, and would like to observe it during the design process.

No participants had a complete initial understanding of the requirements and constraints, which required explicit and implicit task interpretation. This result was expected, considering the number of requirements and constraints. However, all participants understood that they needed to complete each item listed in Table III. They also understood that the problem implicitly required them to organize potential classes "in a logical way" since the classes will "interact in a specific way." Anne, LStew, and Rusty also added that exercising creativity, as directed in the problem, would affect their class design.

In designing the classes, LStew further added that she needed to "avoid common object-oriented programming pitfalls by reducing coupling, reducing interdependencies, and avoiding the diamond of death."Plausibly, this implicit understanding was informed by her interests and experience in OO-design best practices.

All participants considered OO design principles and UML diagram notations as relevant concepts. Rusty and LStew also added that design practices in writing a class diagram and software usability as essential knowledge and skills. Thus, all participants were able to identify relevant concepts to complete the problem correctly.

In order to solve the problem, all participants determined that they need to (1) reread the problem description; (2) identify potential classes; (3) draw the class; (4) establish the classes' relationship; and (5) refine the class diagram as necessary. Interestingly, while the male participants concentrated on rereading the problem description on their first step, the females also concerned with identifying and rewriting the requirements and constraints in their own words. Additionally, Rusty and LStew added that they needed to monitor their progress and address creativity issues throughout their problem-solving enterprise.

18 b) Revising the Initial Task Understanding

The participants executed their problem-solving steps carefully. LStew, for example, started by rereading the problem description and developed a list of requirements. She continued by solving the identified requirements that were related to items, characters, special abilities, player actions, spaces, buildings, players, games, and turn. Sometimes, after completing one of the requirements, she adjusted her design. For example, after designing the action-related classes, she revised the item and character classes. LStew also enhanced her design by making it as logical and as clear as possible so people could easily understand how the classes work together.

When rereading the problem description, the participants were frequently observed as if interpreting it for the first time. These activities were coded as monitoring of task interpretation. Some of these activities triggered them to adjust their task interpretation. Jake, Anne, Rusty, and LStew were observed investing 37.50%, 50.38%, 31.12%, and 36.47% of their engagement for interpreting the task, respectively, including for monitoring and adjustment. Rusty said during the interview, "The general understanding did not really change because I knew that I was going to be creating this class diagram, but as far as the design decisions, it changed a lot."

19 c) Factors that Influence the Task Interpretation Revisions

As mentioned by Rusty, most of the revisedtask interpretations were somehow related to design decisions, such as classes and their behaviors. When addressing each requirement and constraint, the participants need to consider the best mechanism to incorporate it into their existing design. Such need encourages them to reread the problem description as if they encountered it for the first time. This finding aligned with various reports that argued students were required to employ vast cognitive skills and work with different abstraction levels during a programming design activity [45], [46].

All participants except Anne were observed updating their task understanding when addressing creativity requirements. For example, after rereading the third requirement (see Table III), LStew said, "What kind of special instructions could you have if it was a castle versus an inn? I suppose-or a castle versus a fortress? Oh, nothing comes up. Well, a castle can have a king in it, right? ? Okay, so if you are a king and you land on a castle owned by someone else, you get a discount on your rent.” The above illustration showed how LStew’s interpretation of ”specific instruction” evolved as she infused her creativity into the design.

Unlike the other participants, Anne did not attempt to put creativity into her design. Using the third requirement as an example, Anne addressed it by just creating a class called Instructions that would be used by the Space class. At the beginning of solving the problem, Anne commented, ”No one will hire me for my creativity,” suggesting she was not confident of that particular skill.

20 VII.

21 Discussion, Conclusion, and Implication

The analysis results suggested that the participants were competent in identifying the problem goal, requirements, constraints, relevant concepts, relevant experience, and steps to solve an OO design problem. It is important to note that they were able to identify most of it after the initial reading of the problem. However, due to the problem’s extensiveness, they were unable to determine all detailed requirements and constraints.

During the design, they displayed some attributes of expert problem-solvers (see [47], [48]), such as considering possible concerns from various stakeholders. Their awareness of the problem complexity and prior experience in solving OO design problems also inspire a positive behavior; in such, it drove them to be cautious in interpreting the requirements. Thus, it might be beneficial to train students to identify problem characteristics and its complexity as early as possible. Two educational theory may help in this issue, which are Jonassen’s problem types [34]- [36] and Bloom’s Taxonomy to define the problem characteristics.

The analysis results suggested that the participants had a relatively similar approach in solving an OO design problem with extensive requirements and constraints. This approach included rereading the problem description, identifying requirements, identifying classes, determining the classes’ relationships, and refining the class diagram. This finding aligned with various arguments that students developed metacognitive knowledge about the tasks based on their problem-solving experience [1], [49], [50]. Since these metacognitive knowledge influence students’ problem-solving approach [1],it might be beneficial for the instructors to check and ensure that students could acquire that knowledge correctly.

There was self-regulation different between male and female, in such that both female participants listed the requirements and constraints using their own words. However, since all participants unable to identify the requirements and constraints completely, it is impossible to comments more on this difference.

The findings suggested that the participants’ interest and experience influenced their initial and revised-task interpretations. Similarly, when addressing creativity requirements, they also exploited their interest and experience. One study argues that creativity is primarily related to the design process [51]. Thus, Anne’s discomfort about her creative side might be induced by a lack of exposure to a variety of products, and chances to express her creativity. These issues could be fixed by exposing students to various creative software products and encouraging them to tap into their creative side in several programming assignments.

The analysis suggested that task interpretation skills might be deteriorated due to being overwhelmed and drawing from irrelevant experience. This findings also suggested that the participants’ incorrect assumption of educational tasks might affect their selfregulation. Students need to be aware of this potential danger in their education.

22 Conclusion

This study shows that the participants, senior CS students, are capable of drawing explicit and implicit information from an OO-design problem. Most of this information is identified during their initial task interpretation. It is important to note that various contexts influence their task interpretation skills; this is coherent with SRL theory [4], [5], [8], [13], [52] and other existing research [19], [21]. This study shows how participants’ perception of the problem (e.g., domain and complexity) and their experience, interest, and selfefficacy influences their task interpretation (and selfregulation in general). Thus, it is also essential to help students more aware of such contextual information when solving a problem.

This study also shows that participants’ task understanding evolves during their problem-solving endeavor. In terms of solving an OO-design problem, revised-task interpretations are mostly related to design decisions, such as considering the interplay among classes. These senior students also display expert like behaviors where they try to interpolate possible concerns from various stakeholders. All participants also have developed a similar problem-solving approach to OO-design problems. A slight difference exists between males’ and females’ approach, where the females prefer to develop a list of known requirements and constraints.

23 IX.

24 Limitations

This qualitative multiple case study was not designed to produce generalizable results but rather to capture as much variety of students' task interpretation while solving OO-design problems as much as possible. With such a goal, having four participants was adequate for a qualitative case study research [30]. When interpreting the findings, remember that the participants' diversity in this study was limited to their sex. There was a limitation regarding the problem types, such that the research tasks were limited to OO and imperative paradigms. Finally, one study argues that although thinking aloud is commonly used in educational studies, it might also affect students' self-regulation [53] and then influence the research results. Unfortunately, there is no known approach to overcome it.

This paper only focuses on the participants' SR while working on OO design problem. The other unit of analysis is discussed in [21]. [1] [2]

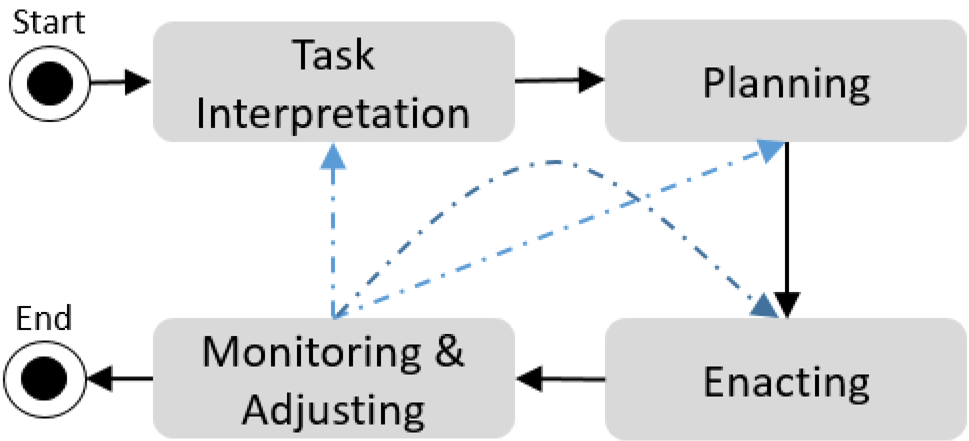

Figure 1:

I

Strategic Action	Definition
Task Interpretation	Students' understanding of the task and associated process to complete it [17]
Planning Strategies	

Figure 2: Table I :

[1] Year 2020 © 2020 Global Journals Students' Understanding of an Object-Oriented Design Task -A Case Study

[2] © 2020 Global JournalsStudents' Understanding of an Object-Oriented Design Task -A Case Study

II

Layer	Question
Explicit	What is the primary goal of this problem?
Explicit & Implicit	In relation to the program that you will design, what are the requirements and constraints that you need to consider?
Implicit	What are the programming concepts related to this problem?
Implicit	What are your previous experiences related to this problem?
Implicit	In relation to the program that you will design, what are the steps (e.g., tasks) that you need to take?

Figure 3: Table II :

[Coll. Rec ()] , *Coll. Rec* 2004. 106 (9) p. .

[Res (2008)] , J Res . 10.3102/0002831207312909. Mar. 2008. 45 p. .

[Improv (2008)] , Q Improv . 10.1111/j.1937-8327.1988.tb00021.x. Oct. 2008. 1 p. .

[First] *Author was born in xx in x. X received the B.S. and M.S. degrees in Computer Science from the X, in X and the Ph.D. degree in X in X*, A First . (Short bio: interests, academic activities, research activities, and professional associations)

[Second] *Author was born in xx in x. X received the B.S. and M.S. degrees in Computer Science from the X, in X and the Ph.D. degree in X in X*, B Second . (Short bio: interests, academic activities, research activities, and professional associations)

[Butler and Cartier ()] 'Case Studies as a Methodological Framework for Studying and Assessing Self-Regulated Learning'. L Butler , S C Cartier . *Handbook of Self-Regulation of Learning and Performance*, D H Schunk, J Greene (ed.) (New York, New York, USA) 2018. Routledge. p. . (2nd ed.)

[Johnson] *Cognitive Analysis of Expert and Novice Troubleshooting Performance*, S D Johnson . (Perform)

[Butler et al. (2015)] 'Collaborative inquiry and distributed agency in educational change: A case study of a multi-level community of inquiry'. D L Butler , L Schnellert , K Macneil . 10.1007/s10833-014-9227-z. *J. Educ. Chang* Feb. 2015. 16 (1) p. .

[Wing (2008)] 'Computational Thinking and Thinking About Computing'. J M Wing . 10.1098/rsta.2008.0118. *Philos. Trans. A. Math. Phys. Eng. Sci* Oct. 2008. 366 (1881) p. .

[Christiaans and Venselaar (2005)] 'Creativity in Design Engineering and the Role of Knowledge: Modelling the Expert'. H Christiaans , K Venselaar . 10.1007/s10798-004-1904-4. *Int. J. Technol. Des. Educ* Jan. 2005. 15 (3) p. .

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

[Graham and Latulipe (2003)] 'CS girls rock: sparking interest in computer science and debunking the stereotypes'. S Graham , C Latulipe . 10.1145/792548.611998. *ACM SIGCSE Bull* Jan. 2003. 35 (1) p. 322.

[Saulnier and Brisson ()] 'Design for Use: A Case Study of an Authentically Impactful Design Experience'. C R Saulnier , J G Brisson . *Int. J. Eng. Educ* 2018. 34 (2B) p. .

[Butler et al. ()] *Developing Self-Regulating Learners*, D L Butler , L Schnellert , N E Perry . 2017. Toronto, ON, Canada: Pearson Education Inc.

[Febrian et al. ()] 'Do Computer Science Students Understand Their Programming Task?-A Case Study of Solving the Josephus Variant Problem'. A Febrian , Lawanto , Oenardi . *Int. Educ. Stud* 2018. 11 (12) .

[Hadwin ()] 'Do your students really understand your assignments?'. A Hadwin . *LTC Curr. Optim. Learn. Environ* 2006. 11 (3) p. .

[Kumar ()] 'Effects of self-regulated learning in programming'. V Kumar . doi: 10.1109/ ICALT. 2005.131. *Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05)*, 2005. p. .

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

[Chi et al. ()] 'Eliciting Self-Explanations Improves Understanding'. M T H Chi , N De Leeuw , M.-H Chiu , C Lavancher . 10.1207/s15516709cog1803_3. *Cogn. Sci* 1994. 18 (3) p. .

[Hadwin et al. ()] 'Examining Student and Instructor Task Perceptions in a Complex Engineering Design Task'. A F Hadwin , M Oshige , M Miller , P Wild . *The Sixth International Conference on Innovation and Practices in Engineering Design and Engineering Education*, 2009.

[Bergin et al. ()] 'Examining the role of self-regulated learning on introductory programming performance'. S Bergin , R Reilly , D Traynor . 10.1145/1089786.1089794. *First International Workshop on Computing Education Research*, 2005. p. .

[Glaser ()] 'Expert knowledge and processes of thinking'. R Glaser . *Enhancing thinking skills in the sciences and mathematics*, D Halpern (ed.) (Ed. Hillsdale, NJ, USA) 1992. Lawrence Erlbaum Associates, Inc. p. .

[Butler and Winne (1995)] 'Feedback and Self-Regulated Learning: A Theoretical Synthesis'. D L Butler , P H Winne . 10.3102/00346543065003245. *Rev. Educ. Res* Jan. 1995. 65 (3) p. .

[Dinsmore et al. ()] 'Focusing the conceptual lens on metacognition, self-regulation, and self-regulated learning'. L Dinsmore , P A Alexander , S M Loughlin . 10.1007/s10648-008-9083-6. *Educ. Psychol. Rev* 2008. 20 (4) p. .

[Falkner et al. ()] 'Gender Gap in Academia: Perceptions of Female Computer Science Academics'. K Falkner , C Szabo , D Michell , A Szorenyi , S Thyer . 10.1145/2729094.2742595. *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education -ITiCSE '15*, (the 2015 ACM Conference on Innovation and Technology in Computer Science Education -ITiCSE '15) 2015. p. .

[Outlay et al. (2017)] ‘Getting IT Together: A Longitudinal Look at Linking Girls’ Interest in IT Careers to Lessons Taught in Middle School Camps’. C N Outlay , A J Platt , K Conroy . doi: 10.1145/ 3068838. *ACM Trans. Comput. Educ* Aug. 2017. 17 (4) p. .

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

[Shaft ()] ‘Helping Programmers Understand Computer Programs: the Use of Metacognition’. T M Shaft . *ACM SIGMIS Database* 1995. 26 (4) p. .

[Hoffman (ed.) ()] *How can expertise be defined? Implications of research from cognitive psychology*, R R Hoffman . Exploring Expertise, R. Williams, W. Faulkner, and J. Fleck (ed.) 1996. Edinburgh, Scotland: University of Edinburgh Press. p. .

[Lewis et al. ()] ‘I Don’t Code All Day’: Fitting in Computer Science When the Stereotypes Don’t Fit’. C M Lewis , R E Anderson , K Yasuhara . 10.1145/2960310.2960332. *Proceedings of the 2016 ACM Conference on International Computing Education Research -ICER ’16*, (the 2016 ACM Conference on International Computing Education Research -ICER ’16) 2016. p. .

[Renumol et al. (2010)] ‘Identification of Cognitive Processes of Effective and Ineffective Students During Computer Programming’. V G Renumol , D Janakiram , S Jayaprakash . 10.1145/1821996.1821998. *ACM Trans. Comput. Educ* Aug. 2010. 10 (3) p. .

[Falkner et al. ()] ‘Identifying computer science self-regulated learning strategies’. K Falkner , R Vivian , N J G Falkner . 10.1145/2591708.2591715. *Proceedings of the 2014 conference on Innovation & technology in computer science education -ITiCSE ’14*, (the 2014 conference on Innovation & technology in computer science education -ITiCSE ’14) 2014. p. .

[Zimmerman] ‘Investigating Self-Regulation and Motivation: Historical Background, Methodological Developments, and Future Prospects’. B J Zimmerman . *Am. Educ*

[Lawanto and Febrian ()] ‘Investigating the Influence of Context on Students’ Self-Regulation during the Capstone Design Course (Accepted)’. O Lawanto , A Febrian . *Int. J. Eng. Educ* 2018.

[Butler and Cartier ()] ‘Learning in varying activities: An explanatory framework and a new evaluation tool founded on a model of self-regulated learning’. D L Butler , S C Cartier . *Annual Conference of the Canadian Society for The Study of Education*, 2004.

[Jonassen ()] *Learning to solve problems: A handbook for designing problem-solving learning environments*, D H Jonassen . 2010. (Routledge)

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

[Flavell ()] ‘Metacognition and cognitive monitoring: A new area of cognitive-developmental inquiry’. J H Flavell . 10.1037/0003-066x.34.10.906. *Am. Psychol* 1979. 34 (10) p. .

[Butler ()] ‘Metacognition and Learning Disabilities’. D L Butler . *Learning About Learning Disabilities*, B Y L Wong, Ed Toronto (ed.) 1998. Academic Press. p. . (2nd ed)

[Flavell ()] ‘Metacognitive Aspects of Problem Solving’. J H Flavell . *The Nature of Intelligence*, L B Resnick (ed.) (Ed. Hillsdale, NJ, USA) 1976. Erlbaum. p. .

[Butler and Cartier ()] ‘Multiple Complementary Methods for Understanding Self-Regulated Learning as Situated in Context’. D L Butler , S C Cartier . *Annual Meeting*, 2005. American Educational Research Association. p. .

[Carver and Scheier ()] ‘Origins and functions of positive and negative affect: A control-process view’. S Carver , M F Scheier . 10.1037/0033-295X.97.1.19. *Psychol. Rev* 1990. 97 (1) p. .

[Lawanto ()] ‘Pattern of Task Interpretation and Self-Regulated Learning Strategies of High School Students and College Freshmen during an Engineering Design Project’. O Lawanto . *J. STEM Educ. Innov. Res* 2013. 14 (4) p. 15.

[Butler and Cartier] *Promoting Effective Task Interpretation as an Important Work Habit: A Key to Successful Teaching and Learning*, D L Butler , S C Cartier . (Teach)

[Butler (1995)] ‘Promoting Strategic Learning by Postsecondary Students with Learning Disabilities’. D L Butler . 10.1177/002221949502800306. *J. Learn. Disabil* Mar. 1995. 28 (3) p. .

[Wang et al. ()] ‘Social Perceptions in Computer Science and Implications for Diverse Students’. J Wang , S Hejazi Moghadam , J Tiffany-Morales . 10.1145/3105726.3106175. *Proceedings of the 2017 ACM Conference on International Computing Education Research -ICER ’17*, (the 2017 ACM Conference on International Computing Education Research -ICER ’17) 2017. p. .

[Alharbi et al. (2012)] ‘Student-Centered Learning Objects to Support the Self-Regulated Learning of Computer Science’. A Alharbi , F Henskens , M Hannaford . 10.4236/ce.2012.326116. *Creat. Educ* Oct. 2012. 03 (06) p. .

[Lawanto and Stewardson (2013)] 'Students' interest and expectancy for success while engaged in analysis-and creative design activities'. O Lawanto , G Stewardson . 10.1007/s10798-011-9175-3. *Int. J. Technol. Des. Educ* May 2013. 23 (2) p. .

[Rivera-Reyes et al. (2017)] 'Students' Task Interpretation and Conceptual Understanding in an Electronics Laboratory'. P Rivera-Reyes , O Lawanto , M L Pate . 10.1109/TE.2017.2689723. *IEEE Trans. Educ* Nov. 2017. 60 (4) p. .

[Lawanto et al. (2018)] 'Students' Task Understanding during Engineering Problem Solving in an Introductory Thermodynamics Course'. O Lawanto , A Minichiello , J Uziak , A Febrian . 10.5539/ies.v11n7p43. *Int. Educ. Stud* Jun. 2018. 11 (7) p. 43.

[Lawanto et al. ()] 'Task Affect and Task Understanding in Engineering Problem-Solving'. O Lawanto , A Minichiello , J Uziak , A Febrian . *J. Technol. Educ* 2018.

[Isomöttönen and Tirronen (2013)] 'Teaching programming by emphasizing self-direction'. V Isomöttönen , V Tirronen . 10.1145/2483710.2483711. *ACM Trans. Comput. Educ* Jun. 2013. 13 (2) p. .

[Havenga ()] 'The Role of Metacognitive Skills in Solving Object-Oriented Programming Problems: a Case Study'. M Havenga . *TD J. Transdiscipl. Res. South. Africa* 2015. 11 (1) p. .

[Abdillah et al. (2016)] 'The Students Decision Making in Solving Discount Problem'. T Abdillah , S Nusantara , H Subanj , A Susanto , Abadyo . 10.5539/ies.v9n7p57. *Int. Educ. Stud* Jun. 2016. 9 (7) p. 57.

[Jonassen (2000)] 'Toward a design theory of problem solving'. H Jonassen . 10.1007/BF02300500. *Educ. Technol. Res. Dev* Dec. 2000. 48 (4) p. .

[Tigerfish] 'Transcription Style Guide'. Tigerfish . *Tigerfish* p. 11.

[Irani (2004)] 'Understanding gender and confidence in CS course culture'. L Irani . 10.1145/1028174.971371. *ACM SIGCSE Bull* Mar. 2004. 36 (1) p. 195.

[Leiviskä and Siponen ()] 'Understanding Why IS Students Drop Out: Toward A Process Theory'. K Leiviskä , M Siponen . *ECIS 2013 Proceedings*, 2013. p. .

[Peng et al. ()] 'Visualizing the Complex Process for Deep Learning with an Authentic Programming Project'. J Peng , M Wang , D Sampson . *Educ. Technol. Soc* 2017. 20 (4) p. .