Turkish Online Journal of Qualitative Inquiry (TOJQI)
Volume 10, Issue 1, January 2019: 1-25
DOI: 10.17569/ tojqi.428953

Research Article

Investigation of Students' Cognitive Processes in Computer Programming: A Cognitive Ethnography Study[1]

Sibel Doğan[2], Orhan Aslan[3], Mehmet Dönmez[4], Soner Yıldırım[5]

Abstract

The aim of the current study is to investigate how cognitive processes of students categorized as novice, semi-expert and expert differ in terms of creating pseudocode for a given programming task. To conduct this aim, cognitive ethnography research design was employed to reveal the cognitive process of the participants behind the specified task. In the study, three undergraduate students from a Computer Education and Instructional Technology (CEIT) department were included as participants. These students were categorized based on two parameters. The first one was the courses that took and the second on was their experiences on programming. While selecting participants, purposeful and snowball sampling methods were used. To collect data, semi-structured interviews, video recording, think aloud procedure, retrospective reviews, observations and document analysis were used. The results showed that participants differed in terms of their decision making and task completion durations, the path they followed, and their perspectives about handling the question.

Keywords: *Cognitive ethnography, programming, pseudocode, cognitive process, coding*

[1] This study was supported by METU-BAP

[2] Res.Asst., Middle East Technical University, Faculty of Education, Department of Computer Education and Instructional Technology, sidogan@metu.edu.tr, https://orcid.org/0000-0002-8688-8578

[3] Res.Asst., Middle East Technical University, Faculty of Education, Department of Computer Education and Instructional Technology, aorhan@metu.edu.tr, https://orcid.org/0000-0003-2513-2010

[4] Res.Asst., Middle East Technical University, Faculty of Education, Department of Computer Education and Instructional Technology, mdonmez@metu.edu.tr, https://orcid.org/0000-0003-0339-5135

[5] Prof.Dr.., Middle East Technical University, Faculty of Education, Department of Computer Education and Instructional Technology, soner@metu.edu.tr, https://orcid.org/0000-0002-3167-2112

Received: 31.05.2018 Accepted: 30.01.2019

Bilgisayar Programlamada Öğrencilerin Zihinsel Süreçlerinin İncelenmesi: Bir Bilişsel Etnografya Çalışması

Öz

Bu çalışmanın amacı, programlama ile ilgili belirlenen bir öğrenme görevinde acemi, yarı uzman ve uzman olarak sınıflandırılan öğrencilerin sözde kod oluşturma esnasındaki zihinsel süreçlerinin nasıl farklılaştığını araştırmaktır. Bu amacı gerçekleştirebilmek için bilişsel etnografya araştırma deseni olarak tercih edilmiştir. Çalışmada, Bilgisayar Eğitimi ve Öğretim Teknolojileri (CEIT) bölümünden üç lisans öğrencisi katılımcı olarak yer almıştır. Bu öğrenciler, öncelikle aldıkları dersler ve programlama konusundaki deneyimlerini göz önünde bulundurarak sınıflandırılmıştır. Katılımcı seçerken, amaçlı ve kartopu örnekleme yöntemleri kullanıldı. Verilerin toplanması için yarı yapılandırılmış görüşmeler, video kaydı, yüksek sesli düşünme, geriye dönük inceleme, gözlem ve doküman analizi kullanılmıştır. Sonuçlar, katılımcıların karar verme ve görev tamamlama süreleri, takip ettikleri yol ve bu soruları ele alma perspektifleri açısından farklılaştıklarını göstermiştir.

Anahtar Sözcükler: Bilişsel etnografya, programlama, sözde kod, zihinsel süreç, kodlama

Introduction

Conceptual Framework: Pseudocode and Cognition

In recent years, programming has become an essential part of everyday life and the concept of programming has become a necessity to survive in this global society. There is a need for acquiring basic competencies such as computer skills, different kinds of media, information technology and information literacy skills (Nelson, 2009). In fact, computational thinking skills is a fundamental skill for programming, and it should be integrated into all levels of the educational curriculum in order to enhance abstract thinking skills (Henderson, Cortina, Hazzan, & Wing, 2007).

According to Papadopoulos and Tegos (2012), it was emphasized for computer science education that students need to be equipped with higher order skills such as problem solving, critical and computational thinking skills. In Turkey, the Information and Communication Technology (ICT) courses have been employed to promote these skills in education. ICT and IT as terms in Turkey can be used interchangeably. Even, ICT courses are redesigned for primary and secondary education (Kalelioglu & Gulbahar, 2014). For example, in Turkey, the coding lessons are now being added to the curriculum of the early stages of education. Moreover, the Ministry of National Education (MoNE) gave ICT teachers the chance to improve it.

According to Özdener (2008), it was stated that students should be required to learn about algorithms and logic behind programming before they start to learn any kind of programming language. In other words, rather than memorizing the commands or codes, learners need to have the ability for establishing the structure such as with algorithms or pseudocode. Algorithm has been defined as a set of precise rules determining how to provide a solution to a problem or to perform a task (Garner, 2006). On the other hand, pseudocode is an alternative way to express complex algorithms by using general wording rather than syntax, keywords and comments in a particular programming language (Garner, 2006). Neither algorithm nor pseudocode depend solely on programming languages. They are used to detail the specific steps that should be followed in order to complete a given task. It might be problematic for students to work with algorithm structures (Milková & Turčáni, 2006; Spohrer & Soloway, 1986), whereas instead, students can be supported to express their thoughts through pseudocode which can help them create connections between concrete, intuitional and symbolic knowledge (Noss, Healy, & Hoyles, 1997).

As a result, computational thinking skills is an important concept for teaching programming. While integrating programming into early levels of education, children' cognitive development processes should be taken into consideration. Computational thinking is a higher order thinking skill and it can be hard to teach it to children in early stage. According to Piaget's stages of cognitive development (Driscoll, 2000), children may not be ready to think in an abstract way before the age of 11. At this point, pseudocode can be utilized as an initial step.

The aim for the current study is to determine how the cognitive processes of university students, categorized as novice, semi-expert and expert, differ while creating pseudocode for a given task. At the end of the study, it is aimed to provide recommendations for students at the lower levels by considering the processes and where participants experienced difficulties.

Methodology

Cognitive ethnography is a method that studies cognition in everyday activities. It is rooted in both ethnography (Williams, 2006) and situated cognition. It combines human behavior in environmental context and task-oriented thinking processes. It was developed by Hutchins (1995b) and is the study of how cognitive activities are completed within the real-life setting (Hutchins, 2003; Williams, 2006). Although rooted in ethnography, cognitive ethnography investigates individual cultures and how individuals create meaning for a phenomenon (Williams, 2006). Moreover, in cognitive ethnography, activities are recorded and analyzed by segmenting unlike interviewing with groups. Furthermore, it deals with the processes such as momentarily activity development rather than focusing on defining commonalities in cultural groups. In other words, cognitive ethnography questions how an activity is accomplished and how knowledge is constructed rather than defining or focusing only on knowledge (Williams, 2006).

In addition, cognitive ethnography involves researchers who are the part of the community or live within it but their purpose is to investigate cognitive processes of participants and context in the community (Dubbels, 2011). The method of cognitive ethnography involves detailed microanalysis of recordings of the cognitive activity consisting of problem solving, decision making and reasoning (Williams, 2012). Moreover, it is conducted with multiple participants with diverse tools and artifacts (Williams, 2012). It is characterized by three main factors (Ball & Ormerod, 2000). First, data are acquired from determined or representative time slices of a situated activity. To exemplify, the segment or the slice in which the decision-making process occurs in is analyzed in detail rather than analyzing a whole video recording (Ball & Ormerod, 2000). Second, cognitive ethnography is purposive because it intends to understand the variation among individual and other external information resources such as other experienced people in terms of strategies used. Third, it emphasizes verifiability of data and method used for data collection by enabling replication of observation with different observers and methodological triangulations (Ball & Ormerod, 2000).

Situated cognition is the theoretical framework for cognitive ethnography (Hutchins, 1995a). It was defined by Robbins and Aydede (2009) as dynamic ongoing interactions with both the physical and sociocultural environment for the purpose of creating ways for human knowledge and understanding. It is a theory implying that thinking is not located in the brain. It is a complex process including different sequential connections in which information is generated from multiple sources among the brain, body and its surroundings (Hutchins, 1991, 1995b; Hutchins & Klausen, 1996).

Cognitive abilities have not evolved for us to understand a complete presentation of the world around us. Rather than focusing on the whole picture, we direct our attention to what is apparent to decide our next step (Merikle, Smilek, & Eastwood, 2001). With this absolute fact, experts differ from novices by having knowledge about where to look for necessary information. Situated cognition emphasizes that our mental processes interact with our body and its surroundings, whether real or virtual. Distributed cognition deals with external interactions between objects, individuals, artifacts, and tools in the environment (Hutchins, 1995a). On the other hand, embodied cognition is interested in the internal processes of the physical body. It was explained that cognition relies on experiences that are a combination of both bodily interactions and cognitive abilities (Thelen, Schöner, Scheier, & Smith, 2001). In the current study, embodied cognition is the focus. In other words, the findings acquired from the performances of participants in the process of creating pseudocodes for the given programming task. Those performances are the act of writing and telling out loud.

The methodology for the current study focuses on cognitive ethnography. For the current study, three students' decision-making processes and cognitive procedures they followed, and how those three students differ in term of their cognitive interactions while performing a given task were investigated. In order to reach these aims, the following research questions are the focus of this study:

1. How does the novice, semi-expert and expert create pseudocode for the given programming task?
2. How does the novice, semi-expert and expert differ in terms of creating pseudocode for the given programming task?

While designing the study, firstly, researchers created the task with the help of subject matter experts (SMEs) who were interviewed about what kind of questions could be asked and what critical points or decisions are relevant to the task. Then, the criteria for the participant selection was decided as well as the participants to be included in the current study by way of snowball sampling. Having decided on the task and the participants, the data collection tools selected were interviews to obtain demographic information, video recordings, retrospective reviews, document analysis and observations performed before, during and after the study.

Task Creation and Participant Selection

To create an appropriate task for the selected students, SMEs were chosen from instructors and assistants of the "Programming Languages I and II" courses. These courses are provided by the CEIT department as compulsory to the undergraduate program. The SMEs were asked for a programming question to be set for the participants during the study. It was decided that the programming question should cover defining variables, using arithmetic operators, conditional statements and loops, and knowledge of extracting data from a file. According to these specifications, a programming question was defined in conjunction and in agreement with the SMEs. The specified question was about calculating the average grades and letter grades from midterm, laboratory and final exam grades of a class of students. After calculations, participants are asked to print the grade mean of the class and a histogram showing the letter grades of the class on the screen as an output of a program. The specified programming question was reviewed by the three researchers of the current study. Then, necessary revisions were applied with the guidance of the SMEs, and a final version of the question was piloted with three students to check the question's clarity and understandability. After setting the question, it was decided to ask the question as independent from any programming language according to the advice of the SMEs. Therefore, the pseudocode of the question was decided to be asked to the study's participants. Then, three of the researchers, who graduated from the CEIT department and have worked as research assistants for five years, created the pseudocodes of the question by themselves. The pseudocodes of each researcher were then compared and combined under a common theme. This theme of the pseudocode was examined by the SMEs in order to create a sample rubric. The pseudocode of the programming question of the study includes the following topics:

- Defining variables (integer, string, array…)
- Getting input from a file
- Using arithmetic operators
- Using conditional statements
    - If, else if
- Using loops
    - For loops
    - Nested for loops
    - While loops
    - Do while loops

Furthermore, three students with similar background to the study's participants were asked to create their own pseudocode of the programming question in order to check the applicability of the question before applying it to the actual participants of the study. Any necessary revisions were completed in accordance with their comments. Following these reviews and updates, the final version of the programming question was ready to implement with the study's participants who are novice, semi-expert and expert students from the CEIT department. The aim of this study was to determine how cognitive processes of students categorized as novice, semi-expert and expert differ in terms of creating pseudocode for a defined programming task. To manage this aim, selecting participants and defining criteria for novice, semi-expert and expert was crucial because they contribute to the phenomenon. In selecting participants, two sampling methods; criteria-based and snowball method were used.

Firstly, SMEs were asked about student profile and criteria for defining novice, semi-expert and expert. After negotiations, some criteria related with the courses students took were determined. Thus, students who took “Programming Language I (CEIT210)” course were categorized as novice, students who took both the “CEIT210” and “Programming Language II (CEIT211)” courses were categorized as semi-expert, and students who took the “CEIT210”, “CEIT211” and “Project Development and Management (CEIT435)” courses were classified as expert. In addition, expert students were expected to write software by using any kind of programming language besides their formal education. In other words, it was aimed to find a person interested in programming intuitively.

After defining criteria, snowball sampling was used to select the participants. Firstly, SMEs were asked for suggestions. Both researchers and SMEs built consensus by negotiating about recommendations. In this process, the role of the researchers was also important as they have close relationship with the students because they provide assistance for the courses. After determining students for each category, the students were then interviewed in order to obtain demographic and background information. Detailed information was presented for each student in Table 1.

Table 1
Demographic Information of Participants

	Novice	Semi-Expert	Expert
High School	Vocational High School Web design and programming department	Vocational High School Web design and programming department	Vocational High School Web design and programming department
Grade level	Sophomore	Junior	Senior
Criteria	CEIT210	CEIT210 CEIT211	CEIT210 CEIT211 CEIT 435 “IEEExtreme Algorithm contest”

Participants of the study had similar education background. All of them graduated from vocational high school and their departments were web-design and programming. Also, all subjects were students at the same university and same department. Moreover, both novice and semi-expert started to be interested in programming in high school but the expert interested in programming since twelve years old. Thus, it can be said that the main difference between them was related with their programming experiences.

The determined courses were important in terms of categorizing students as novice, semi-expert and expert. In the Programming Language I and II courses, students obtain experience with programming logic and have the chance to develop their knowledge to some degree. However, the students actually get to apply their experience in the Project Management course, in which students produce a comprehensive project by using any kind of programming language.

Data Collection Instruments

In this part of the study, different types of data sources, namely semi-structured interviews with SMEs and participants, video recording, observations, think aloud procedure, document analysis and retrospective review were used. These tools were conducted before, during and after the study. Semi-structured interviews were used before starting the study. Observation, think aloud procedure and video recording were performed during the study. Finally, retrospective review was carried out after the study was completed. Afterwards, these tools were explained in detail.

Firstly, before collecting data from the participants, semi-structured interviews were conducted with SMEs to decide on the task and the participants to be included in the study. There were two SME's; one is programming languages instructor and the other is the research assistant helping the lecturer with the course. After providing brief information about the research to the SME's, they were asked questions about what kind of a task can be given to students that requires them to think, reflect and demonstrate differences in terms of expertise. Moreover, they were requested to provide their opinions about the kinds of prior knowledge participants need to have, performance standards for the task, as well as criteria to select participants as novice, semi-expert and expert.

Having collected data from the SMEs and decided on the task and participants to be included in the study, semi-structured interviews were conducted with the participants. In the interviews, participants were asked to provide information about themselves regarding the type of high school they had graduated from, the first time they learned algorithm, whether or not they had developed any software, and their background in programming.

The second data collection tool was video recording. While participants were performing the given task, a video camera recorded the participant undertaking the task. Video recording is important in cognitive ethnography studies because it provides chance to review the cognitive process aimed to be analyzed in detail multiple times back and forward. The third tool was think aloud procedure that the participants used. This tool was included in the study to support and empower the points highlighted by the participants and their work (pseudocode). As the fourth tool, while the participants were completing the given task, researchers took observation notes based on participants' problems, comments and questions related to the task, as well as further questions that will be asked in retrospective review.

After completing the task, the fifth tool, retrospective review was completed in which the researchers and each participant reviewed their respective video recording. During the review, participants were asked to reflect why they followed a certain way and what they thought while doing it. Last, document analysis was conducted on the participant's work (pseudocode). Their work was analyzed based on the predetermined rubric created by the researchers and the SMEs to investigate the performance and processes followed by the participants.

Data Analysis

Before conducting the study, firstly, semi-structured interviews were conducted with SMEs and the participants. These interviews were transcribed word by word and member checked by researchers. The interview results of the SMEs were used while selecting participants and creating the task in order to define the main phases and steps of them. The task was divided into three main phases, namely preparing the data, coding, and printing outputs. Moreover, steps that participants supposed to conduct were determined for each phase.

Secondly, researchers took observation notes and their notes were then compared and contrasted with each other in order to scrutinize their different perspectives. During the observation, notes were taken about the participants' think aloud procedure and performance. Thirdly, having just completed pseudocodes, retrospective review was performed with each participant by watching video recording together. During the retrospective review, the major points from the observation notes based on both researchers notes and think aloud data were discussed in detail. The aim was to make them think about their performances and provide deeper information about their cognitive processes. Then, recordings from retrospective review were analyzed considering each phase and steps. In this process, participants' explanations about their performances were transcribed word by word.

Fourthly, document analysis was applied to the final version of each participant's pseudocode. The researchers analyzed the participants' solution to determine how they come up with a particular solution and investigate how they differentiate from each other. Then, the three researchers' analysis for each phase and steps were compared to be sure about the accuracy of the participants' work. Lastly, participants were recorded with a video camera from behind while they conducted their task. The obtained video record was divided into part according to determined phases and steps. Each part was reviewed multiple times to get deeper knowledge about participants' performances and understand their cognitive processes. Moreover, while analyzing these parts, data obtained from observation notes, retrospective review and document analysis were considered.

Researcher Role

In the current study, researchers had an insider status. They had active roles in designing the task applied to the participants, and in determining the criteria for selecting and categorizing participants as novice, semi-expert and expert under the supervision of SMEs. Moreover, they were actively involved in the process of revealing how students' cognitive processes differ while creating the pseudocode for the programming task. The researchers' educational background is suitable for this study. All three researchers graduated from the department of Computer Education and Instructional Technology (CEIT). They are also conducting their PhD studies within the department. Moreover, three of the researchers work as research assistants at the CEIT department and assist with courses. Moreover, they work with students Thus, they have information about the courses and their content. They all have software backgrounds at different levels.

Quality of the Research

In the current study, several measures were taken to ensure the validity issues. Firstly, member check was used (Merriam, 1998). The prepared question and rubric pseudocode were checked by two SMEs. Mathison (1988) explained that it is expected to use different data sources and tools to triangulate data. Triangulation is a strategy improving validity of the research (Huberman & Miles, 1994). In the current study, different data tools of video recording, think aloud procedure, retrospective review, observation and document analysis were used to collect data from the same participants. Furthermore, the designed programming task was piloted with three students to check the clarity of the task and was revised by considering misconceptions and grammatical errors. For further validation, while conducting the task, the students' works observed by researchers in order to perform a crosscheck. Moreover, researchers analyzed pseudocodes of participants separately and then compared their findings. In addition, findings from qualitative studies are less generalizable to other studies (Johnson, 1997), but this can be overcome by providing detailed information about research design. In the current study, the results cannot be generalized because of small sample size. However, the design of the study, task creation, participant selection and data collection procedure as well as their analyses were explained in detail to provide guidance for other researchers.

Findings

Research Question 1: How pseudocode is created for given programming tasks by novice, semi-expert and expert participants?

In this part, participants' cognitive tasks were provided by using a table acquired from video analysis. Their decision making and task completion durations were given in seconds for each steps of the phases; preparing data, coding and printing output.

In the first phase, participants were expected to define variables such as integer, string, and array, and assign variables and read data from the file. For the coding phase, participants were expected to conduct operations such as calculating mean for each individual and for the class, find the corresponding letter grades for each student and count the number of each letter grades. For the last phase, participants were asked to print a histogram for the letter grades and the mean of the class on to the screen. In Figure 1, the processes that should be followed is presented in order to provide an overall framework.

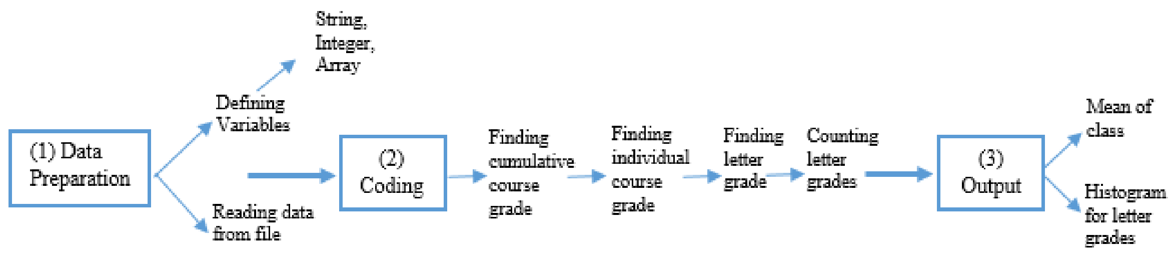

Figure 1. The overall framework of the study

Novice

The task that was given to the "novice" participant had three main phases; with individual steps within some phases. The novice spent 10 minutes completing the whole task. Table 2 shows the decision making and completion durations for the novice participant.

Table 2
Decision Making and Completion Durations – Novice

Phases		Completion duration (seconds)	Decision making duration (seconds)
Data preparation		22	13
Coding	Finding cumulative course grade	194	72
	Finding individual course grade	60	30
	Finding letter grades	101	23
	Counting letter grades	102	102
Printing outputs		15	15

It was highlighted that the novice spent 22 seconds in the first phase, data preparation and deciding to read data from a file. According to document analysis, he just read the data from a file and did not define any variable or decide on what kind of variables would be used in the following parts. Think aloud procedure and observation notes revealed that the novice talked about variables during whole his working time, yet did not demonstrate them in the pseudocode. According to retrospective review, the novice explained this situation in retrospect by saying:

> *"I did not write because I had defined those [variables] in my mind."*

The second phase was the coding, which consisted of four steps. For the first step, finding the cumulative course grade, the novice spent 194 seconds to complete it. Document analysis showed that he did not use any kind of loop for making calculations for 20 people at the same time; rather he preferred calculating the grades of each student by hand sequentially. Moreover, he did not divide the accumulated grades by the number of students to find the mean. Think aloud procedure showed that he knows that an array keeps multiple data, but he did not know how to apply it. He explained it in retrospective review by saying:

> *"I would either define variables one by one or use array. Using array was the easiest way but I thought that I would not be able to apply it [array]. I defined variables one by one because it was the guaranteed way for me."*

Observation notes also revealed that the novice lacked adequate knowledge about the implementation of control statements.

The second step was finding the individual course grade. The novice used 60 seconds for the task to be completed. Document analysis showed that the novice did not use any kind of loop for finding the individual grades. Instead, he went for defining variables and making calculations by hand for each student. During this step, he did not mention using any kind of loop to perform the operations just once. Observation notes also provided information that the novice did not process multiple operations once, but designed the process for one entity at a time. The third step of the second phase was finding the letter grades of the students. The task took 101 seconds for novice to complete. Think aloud procedure indicated that he mentioned about using conditional statement to find the letter grades, but did not apply it to the pseudocode. For the last step, the novice spent 102 seconds. While conducting the step, he put into words using a counter for counting the number of letter grades. However, according to document analysis, the novice did not apply what he had decided for the step. Observation notes revealed that he could build the logic behind the operation, but could not apply it.

The last phase of the study was providing outputs and the novice took 15 seconds in total. According to document analysis, he could not print mean for the class nor the histogram for the number of each letter grades. In the retrospective review, he expressed this by saying:

> *"I could not connect the number of letter grades and printing them as stars [histogram]."*

To conclude, while writing the pseudocode, the novice generally had required preliminary knowledge about what to do. However, he had trouble in the implementation of control statements and variable definitions because of inadequacy of practice making a single operation for multiple entities.

Semi-expert

The semi-expert spent about 15 minutes to complete the whole task. Table 3 shows the decision making and completion durations for the semi-expert participant.

Table 3
Decision Making and Completion Durations – Semi-Expert

Phases		Completion duration (seconds)	Decision making duration (seconds)
Data preparation		75	7
Coding	Finding cumulative course grade	117	10
	Finding individual course grade	68	9
	Finding letter grades	103	47
	Counting letter grades	101	10
Printing outputs		256	25

In the first phase, the semi-expert spent 75 seconds to complete the phase. Document analysis revealed that she read data from a file using a loop, but did not define any variables beforehand. In the retrospective review, she explained the reason by saying:

> *"I realized that I defined variables in my mind and I forgot to write them [variables] down in my pseudocode."*

In the second phase, she performed all four steps in a different way. Document analysis revealed that she first calculated each students' individual grades, then she found the letter grades and counted them. Finally, she reported the cumulative course grade by summing the individual course grades.

For the first step, while trying to find the individual course grades, the semi-expert used 68 seconds to complete the step. Document analysis presented that she used a loop to calculate the individual course grades. During the step, she explained she had trouble in making calculations at first, then decided to use a loop for that purpose.

Afterwards, she combined the second and third steps. Document analysis revealed that she found the letter grades and the number of them in the same loop by using conditional statements. In these steps, she spent 103 seconds to find the letter grades and decide to use control statements. Moreover, the semi-expert allocated 101 seconds to count the number of letter grades and decide to use counters in her conditional statements. Observation notes and her expressions during these steps indicated that she experienced some confusion. Although she was expected to use an array to assign letter grades, she could not, and explained this situation in retrospective review by expressing:

> *"The variables [letter grades] that I assigned kept being overwritten and I could not think how to control [assigning them to variables] it at that moment… I did not think about using array; if I had, it would have made more sense."*

For the fourth step, she spent 117 seconds to complete. Document analysis showed that the semi-expert used a loop to calculate the cumulative course grade by summing the individual course grades and then dividing it by the number of students at the same time.

The last phase was printing outputs. The semi-expert spent 256 seconds to complete and decide to use nested loops. However, document analysis showed that she did not reflect what she thought in the pseudocode. She explained it in retrospective review by saying:

> *"I created it [nested loops] in my mind but I could not verbalize it in my pseudocode. I could have stated it in a clearer sentence."*

To sum up, the semi-expert established the overall structure for the given task and thought of the task as a whole. Although, she experienced confusion in some steps of the task, she overcame this through her own solutions to some point. The semi-expert could not reach the next level due to her lack of content knowledge.

Expert

The expert spent about 27 minutes to complete the whole task. Table 4 presents the decision making and completion durations for the expert participant.

Table 4
Decision Making and Completion Durations – Expert

Phases		Completion duration (seconds)	Decision making duration (seconds)
Data preparation		355	63
Coding	Finding cumulative course grade	452	53
	Finding individual course grade	258	15
	Finding letter grades	133	15
	Counting letter grades	116	39
Printing outputs		90	9

In the first phase, the expert spent 355 seconds for completing the task and decision making. Document analysis showed that the expert conducted this phase in detail. He read data from a file by importing necessary libraries. He transferred the data from the file to a two-dimensional array after first checking whether or not the file was empty. He then prepared the data for further operations. Observation notes of the researchers revealed that the expert planned this phase intensively by considering further steps. While performing the phase, he explained each of his processes in detail by providing rationales behind his thoughts. During the phase, he pointed out different alternatives such as object, array and class.

In the second phase, his first step was to calculate the cumulative course grade. The expert spent 452 seconds in total. Document analysis reported that he defined the function to find the mean of each exam by using the indexes of the pre-defined array. Then, he summed them up to find the course mean. During the process, he explained that he preferred function to avoid making multiple mean calculations.

Having completed the first step, the expert combined the second and third steps. In the second step, it took 133 seconds for expert to use a control statement to calculate the individual course grades. For the third step, he spent 116 seconds to complete and find the letter grades by using conditional statements. Document analysis revealed that the expert defined an array to store the letter grades and a variable for individual grades. In a loop, he calculated the individual grades and assigned them to the result temporarily. Then, with a conditional statement, he equated the individual grades with the letter grades and assigned them to the pre-defined array. Both think aloud procedure and observers' notes demonstrated that his content knowledge was sufficient. He explained all his work with the reasons behind his actions and compared the way he chose with alternative options. In the retrospective review he emphasized this by stating:

> *"While writing code, I always think about alternative methods and try to choose the more efficient course for my work... For example, while defining variables, I think about different types and try to choose one which utilizes the least memory by considering the needs of the program I am writing."*

For the last step of the coding phase, the expert spent 116 seconds for the task. Document analysis indicated that he defined an array to keep he numbers of each letter grade. Moreover, he preferred to use a loop to bring the letter grades to the stage and use a conditional statement to increment counters for each of them. Observation notes indicated that he overlooked that he could combine this step with second and third steps. It would have been an easier way to conduct the operation. In the retrospective review, he explained this by expressing:

> *"I could have included this operation [counting the letter grades] in the previous loop [loop used in second and third steps], but I did not pay attention to this because it was not a complex project."*

Printing outputs was the last phase of the task. It took the expert 90 seconds to complete and print the mean of the class and the histogram. Document analysis showed that the expert ignored printing the mean. He used nested loops for printing the histogram as output. Think aloud procedure indicated that he first thought one loop would be enough, but then realized immediately that he should use nested loops.

In conclusion, the expert's content knowledge was adequate to conduct the task. He explained all his work in detail with reasons and provided alternative methods during the whole task. He built the main structure at the beginning very well. He determined what he need basically and added some other variable definitions during his work. He rectified his mistakes and tried to apply more efficient alternatives during his work.

Research Question 2: How does the novice, semi-expert and expert differ in terms of creating pseudocode for the given programming task?

When the participant's pseudocode was examined, the researchers noted how the participants conducted the task given in the first part of the result. In Table 5, their differentiations are presented for each phase and step in detail.

Table 5
Comparison of Duration – Novice, Semi-expert & Expert

	Data Preparation		Coding		Printing Outputs	
	CD	DMD	CD	DMD	CD	DMD
Novice	22	13	457	227	15	15
Semi-expert	75	7	389	86	256	25
Expert	355	63	959	122	90	9

Note. CD: Completion Duration (seconds), DMD: Decision Making Duration (seconds)

For the first phase, preparing the data, the expert participant spent more time on variable definitions and decisions for further steps. Moreover, he created the structure of the whole task at the beginning. On the other hand, the novice and the semi-expert did not spend as much time and effort on this phase. They just read the data from a file without defining any variable. In the retrospective review, they explained their reasoning as they had just defined variables in their mind.

In the second phase, the participants were supposed to perform operations for calculating the mean for each individual and for the course, finding letter grades and counting the number of grades. During this phase, the novice could not conduct operations for all of the students at once. He could not actualize the multiple operation in a loop. On the other hand, both the semi-expert and the expert considered operations for all the students at the same time by utilizing a loop. Furthermore, in this phase, only the expert thought about storing the calculated results in an array. The novice did not even think about it. However, the semi-expert realized that the results were overlapping, but she was confused about how to resolve the problem. Moreover, during these stage, the expert talked about alternative ways for operations and storing the results. He explained why he chose the method he applied with reasons. He came up with more efficient ways. In addition to these, there were differences in terms of their sequence of calculations. Both the novice and the expert calculated the cumulative course grade before calculating the individual course grades. In the retrospective review, they explained that they divided the task into two parts by considering the problem. In retrospective review, both the novice and the expert stated that they had not understood the question completely. They said that they could have performed better if they handled the question as a whole. However, the semi-expert did consider the problem as a whole. She first calculated the individual course grades and then summed them to find the cumulative course grade. Besides, all the participants thought about calculating individual course grades and finding letter grades in the same loop. However, only the novice could not apply it his pseudocode. Apart from these, the expert used function rather than repetitively calculate the mean over and over.

The last phase was printing the mean of the class and a histogram for the letter grades. Generally, all participants knew what to do for this step. However, the novice could not actualize how to apply the necessary operation. The semi-expert completed this phase successfully, but the expert forgot to print the mean of the class.

Discussion and Conclusion

The results showed that participants differed in terms of the durations they spent, the paths they followed and their perspectives about handling the question they were tasked with. Initially, they differed in terms of the decision making and task completion durations for the steps. The ‘expert’ participant spent more time (~27 minutes) to complete the given task, while the ‘novice’ spent the least amount of time (~10 mins). The reason for this differential was that the expert’s solution was very detailed and included a lot of information as well as alternate solutions. The expert’s extensive knowledge caused him some level of confusion as he could not effectively orchestrate his knowledge for the solution. On the other hand, the solution of the novice participant was superficial.

Next, the participants differed with regards to the path they followed in creating their pseudocode. This differential resulted from the degree as to how well they understood the question. Both the novice and the expert did not visualize the question as a whole, but rather they divided the question in two parts which actually made their job harder. Thus, it can be said that comprehending the question can be thought of as the baseline before starting or creating pseudocode.

Moreover, participants’ perspectives about handling the question were also different. The expert considered the question as a project, the semi-expert as a program, while the novice approached the question like a mathematical problem. Differences in their knowledge levels and amount of practice they’d experienced led to different approaches for the given question. The novice had some knowledge of arrays, but lacked adequate practice in its implementation. As a result, he handled the question as a mathematical problem without considering how a computer will understand written pseudocode that lacked computational thinking.

Coding or programming needs higher level skills such as problem solving, critical and computational thinking (Papadopoulos & Tegos, 2012); skills which require abstract thinking. However, Driscoll (2000) claimed that when Piaget’s stages of cognitive development are considered, children may not be able to develop abstract thinking until the age of eleven. Therefore, until they are able to develop higher level skills, younger students can be supported to express their thoughts as a baseline. Debate classes and communities can be examples of leading students to acquire critical thinking skills. After students are equipped with this skillset, they will be better prepared to transfer their thoughts in an ordered way more easily. Students need to be taught about creating stepwise solutions for a given problem rather than simply diving into the codes, commands and scripts of a specific programming language. Afterwards, more complex and structured tasks or questions can be provided to enhance their problem solving and critical thinking skills. Finally, students can integrate or use such skills within any programming language, and thereby be equipped with a much-enhanced skillset to tackle the tasks they are likely to encounter throughout their educational career and beyond.

References

Ball, L.J., & Ormerod, T.C. (2000). Putting ethnography to work: The case for a cognitive ethnography of design. International Journal of Human-Computer Studies, *53*(1), 147–168. http://doi.org/10.1006/ijhc.2000.0372

Driscoll, M.P. (2000). Psychology of learning for instruction. Allyn & Bacon: Needham Heights, MA.

Dubbels, B. (2011). Cognitive ethnography: A methodology for measure and analysis of learning for game studies. International Journal of Gaming and Computer-Mediated Simulations (IJGCMS), *3*(1), 68–78.

Garner, S. (2006). The development, use and evaluation of a program design tool in the learning and teaching of software development. Issues in Informing Science and Information Technology, 3, 253–260.

Henderson, P.B., Cortina, T.J., Hazzan O. & Wing, J.M. (2007). Computational thinking. In ACM SIGCSE Bulletin, *39*(1), 195-196.

Huberman, A.M., & Miles, M.B. (1994). Data Management and Analysis Methods. In N.K. Denzin & Y.S. Lincoln (Eds.), Handbook of Qualitative Research (pp. 428-444). Thousand Oaks: Sage Publications.

Hutchins, E. (1991). The social organization of distributed cognition. In L.B. Resnick, J.M. Levine, & S.D. Teasley (Eds.), Perspectives on socially shared cognition (pp. 283–307). Washington, DC: American Psychological Association. http://doi.org/10.1037/10096-012.

Hutchins, E. (1995a). Cognition in the Wild. Cambridge, MA: MIT Press. http://doi.org/10.1023/A:1008642111457

Hutchins, E. (1995b). How a cockpit remembers its speeds. Cognitive Science, *19*(3), 265–288. http://doi.org/10.1016/0364-0213(95)90020-9

Hutchins, E. (2003, July). Cognitive ethnography (Plenary address). In R. Alterman, & D. Kirsh (Chairs), 25th meeting of the Cognitive Science Society, Boston.

Hutchins, E., & Klausen, T. (1996). Distributed Cognition in an Airline Cockpit. In Y. Engeström & D. Middleton (Eds.), Cognition and Communication at Work (pp. 15-34). Cambridge: Cambridge University Press. http://doi.org/http://dx.doi.org/10.1017/CBO9781139174077.002

Johnson, R.B. (1997). Examining the validity structure of qualitative research. Education, 118(2), 282-292.

Kalelioglu, F., & Gulbahar, Y. (2014). The effects of teaching programming via Scratch on problem solving skills: A discussion from learners' perspective. Informatics in Education-An International Journal, *13*(1), 33–50.

Mathison, S. (1988). Why triangulate ? American Educational Research Association, *17*(2), 13–17. http://doi.org/10.2307/1174583

Merikle, P.M., Smilek, D., & Eastwood, J.D. (2001). Perception without awareness: Perspectives from cognitive psychology. Cognition, *79*(1-2), 115-134. http://doi.org/10.1016/S0010-0277(00)00126-8

Merriam, S.B. (1998). Qualitative research and case study applications in education. Dados (2nd ed.). San Francisco: Jossey-Bass Publishers. http://doi.org/10.1017/CBO9781107415324.004

Milková, E., & Turčáni, M. (2006). Digital objects supporting development of algorithmic thinking. In A. Méndez-Vilas, A. Solano Martín, J. Mesa González, & J. A. Mesa González (Eds.), Current Developments in Technology-Assisted Education (pp. 376–380. Badajos, Spain, Formatex

Nelson, J. (2009). Celebrating Scratch in libraries: Creation software helps young people develop 21st century literacy skills. School Library Journal. Retrieved 12, April, 2018, from http://www.slj.com/2009/05/opinion/the-gaming-life/celebrating-scratch-in-libraries-the-gaming-life/#_

Noss, R., Healy, L., & Hoyles, C. (1997). The construction of mathematical meanings: connecting the visual with the symbolic. Educational Studies in Mathematics, *33*(2), 202–233. http://doi.org/10.1023/A:1002943821419

Özdener, N. (2008). A comparison of the misconceptions about the time-efficiency of algorithms by various profiles of computer-programming students. Computers and Education, *51*(3), 1094–1102. http://doi.org/10.1016/j.compedu.2007.10.008

Papadopoulos, Y., & Tegos, S. (2012). Using microworlds to introduce programming to novices. In Proceedings of the 2012 16th Panhellenic Conference on Informatics, PCI 2012 (pp. 180–185). Institute of Electrical and Electronics Engineers (IEEE). http://doi.org/10.1109/PCi.2012.18

Robbins, P., & Aydede, M. (2009). The Cambridge Handbook of Situated Cognition. Cambridge: Cambridge University Press. http://doi.org/10.1017/CBO9780511816826

Spohrer, J.C., & Soloway, E. (1986). Novice mistakes: are the folk wisdoms correct? Communications of the ACM, *29*(7), 624–632. http://doi.org/10.1145/6138.6145

Thelen, E., Schöner, G., Scheier, C., & Smith, L.B. (2001). The dynamics of embodiment: A field theory of infant perseverative reaching. Behavioral and Brain Sciences, *24*(1), 1–34.

Williams, R. F. (2006). Using cognitive ethnography to study instruction. In S.A. Barab, K.E. Hay, & D.T. Hickey (Eds.), Proceedings of the 7th International Conference on Learning Sciences (pp. 838–844). International Society of the Learning Sciences. Retrieved 15, January, 2018, from http://dl.acm.org/citation.cfm?id=1150156

Williams, R.F. (2012). Image Schemas in Clock-Reading: Latent Errors and Emerging Expertise. Journal of the Learning Sciences, *21*(2), 216–246. http://doi.org/10.1080/10508406.2011.553259