Journal of Learning and Teaching in Digital Age, 2024, **9**(2), 111-128
https://dergipark.org.tr/en/pub/joltida
ISSN: 2458-8350 (online)

Research Paper

Sustaining Undergraduate Students' Metacognitive Regulatory Actions During Online Flipped Programming Course

Gamze Türkmen[a], Sinan Hopcan[*b] , Elif Polat[c]

[a](ORCID ID: 0000-0002-4695-9159), Manisa Celal Bayar University, Faculty of Education, Computer Education and Instructional Technology, Manisa, Turkiye, gamze.turkmen@cbu.edu.tr
[b](ORCID ID: 0000-0001-8911-3463), İstanbul University-Cerrahpaşa, Hasan Ali Yücel Faculty of Education, Computer Education and Instructional Technology, İstanbul, Turkiye, sinan.hopcan@iuc.edu.tr
[c](ORCID ID: 0000-0002-6086-9002), İstanbul University-Cerrahpaşa, Hasan Ali Yücel Faculty of Education, Computer Education and Instructional Technology, İstanbul, Turkiye, elif.polat@iuc.edu.tr
[*]Corresponding author

ARTICLE INFO
Received: 15 November 2023
Revised: 16 April 2024
Accepted: 26 April 2024

Keywords:
Programming instruction
Metacognitive regulation
Flipped classroom
Flipped programming learning

doi: 10.53850/joltida.1391039

ABSTRACT
This research explores how metacognitive strategies influence the metacognitive awareness of undergraduate students enrolled in an online flipped programming course. It specifically focuses on regulatory actions crucial for success in programming instruction and distance education settings. The primary objective is to contribute to the existing literature by investigating the implementation of online flipped programming courses that integrate metacognitive-oriented approaches to support students' metacognitive regulatory actions. The study employed an explanatory sequential mixed methods design. A total of 29 university students enrolled in programming courses participated in the study, engaging with instructional videos provided before each 10-week lesson. They were administered the Metacognitive Awareness Scale and supplementary forms designed to assess their metacognitive awareness and regulatory actions. A detailed coding scheme was developed to analyze students' metacognitive regulation activities during programming lessons. The study also evaluated the impact of supportive activities on students' metacognitive awareness. While no statistically significant difference was found in the students' metacognitive awareness through quantitative analysis, qualitative data revealed that activities supporting metacognition significantly enhanced students' comprehension of the programming content.

INTRODUCTION

In academic discourse, learning and teaching programming is acknowledged as a challenging field due to the structural complexity inherent in programming. Despite a high level of interest and motivation among students to learn programming, many either drop out or struggle to pass. Programming demands strong problem-solving skills, and, as Mayer (1998) emphasized, it is insufficient for students to merely understand "what" to do in solving routine, new, or previously challenging problems. They must also grasp "when" to apply specific problem-solving strategies. Furthermore, programming problem-solving involves cognitive processes where students must know which strategies to employ and how to use them effectively (Bernard & Bachu, 2015). This underscores the importance of metacognition, defined by Anderson (2002) as the ability to reflect on one's own cognitive processes. This complex concept encompasses an individual's thought processes and self-regulation during problem-solving (Schoenfeld, 2016), making metacognition a critical higher-order thinking skill that enables students to understand, analyze, and evaluate their cognitive processes (Bernard & Bachu, 2015).

Programming skills, as described by Sun et al. (2022), are multifaceted and include metacognition, deemed essential. Metacognitive skills—planning, self-monitoring, and evaluation—are crucial for computer programming, a problem-solving activity (Avcı, 2022). Research suggests that acquiring metacognitive skills during programming instruction can enhance students' problem-solving abilities and performance (Scherer et al., 2020). Hence, programming presents a significant challenge to novice learners, who must develop not only coding skills but also metacognitive abilities (Nurulain Mohd Rum & Zolkepli, 2018). The lack of metacognitive skills can hinder programming learning (Pea et al., 1987), making the development of these skills vital for training competent programmers.

Reflecting on one's programming knowledge and skills, metacognitive skills are essential for achieving expertise in programming. Through metacognition, individuals can identify and correct errors, seek assistance, and program more effectively (Pea et al., 1987). Thus, it is crucial for students to reflect on their programming strategies and critically assess their approaches (Çakıroğlu & Er,

2020), and to make connections between concepts (Bernard & Bachu, 2015). Utilizing error messages, feedback, and related tools in programming enhances metacognitive skills and awareness (Çakıroğlu & Er, 2020). Loksa (2020) highlighted the scant research on fostering metacognitive awareness in programming instruction. Metacognitive skills provide students with effective problem-solving strategies (Anderson, 2002). Educators across various fields, including programming and mathematics, employ metacognitive strategies to help students organize their learning through planning, monitoring, and evaluation (Tsai et al., 2022). Employing these strategies supports struggling students, enabling them to learn more deeply and achieve greater success (Anderson, 2002). This research aims to help students understand how to apply the basic concepts learned in programming lessons to create algorithms and solve problems by focusing on metacognitive regulation stages such as planning, monitoring, and evaluating.

Metacognitive skills are also deemed essential in online learning environments (Zhao & Ye, 2020). Implementing metacognitive strategies in online learning has yielded positive outcomes (Broadbent & Poon, 2015; Karatas & Arpaci, 2021; Yılmaz & Keser, 2017; Zhao & Ye, 2020). As online learning gains popularity, researchers have adopted the "online flipped classroom" model, building on the success of traditional flipped classrooms (Polat et al., 2022). This model involves students learning core content independently before class through pre-recorded videos and/or lecture notes (Clark et al., 2022). The online flipped classroom also requires students to complete pre-class activities asynchronously, such as participating in discussion forums, watching videos, and taking quizzes, with the primary difference being the virtual rather than physical interaction between students and instructors (Jia et al., 2023). In this model, students must be aware of their own knowledge and skills and capable of planning, monitoring, and evaluating their learning progress (Yılmaz & Baydas, 2017).

Purpose of The Study

Stöhr et al. (2020) observed a growing interest in online flipped classrooms, despite limited studies evaluating this approach. While previous studies on metacognition in programming instruction exist (Atmatzidou et al., 2018; Bergin et al., 2005; Cetin et al., 2014; Çakıroğlu & Er, 2020; Loksa et al., 2016; Wang, 2019; Yıldız Durak & Uslu, 2022; Zhou et al., 2021), few have explored metacognitive strategies within the online flipped classroom context. This investigation aims to address the gap in knowledge regarding the impact of metacognitive strategies on undergraduate students' metacognitive awareness scores in an online flipped programming course. It focuses on metacognitive regulatory actions critical to success in programming instruction and online learning environments, examining how students apply metacognitive strategies to complete algorithm assignments. This study's significance lies in its contribution to the literature on implementing an online flipped programming course that integrates metacognitive strategies to promote students' metacognitive regulatory actions.

BACKGROUND

Metacognition in Definition

Metacognition is the knowledge and understanding of cognitive processes, specifically an individual's ability to think, comprehend, and manage their own learning (Schraw & Dennison, 1991: p. 460). It encompasses an individual's awareness of their own cognitive control, including the evaluation of the effectiveness of past strategies used, the monitoring of self-efficacy, and the assessment of metacognitive knowledge based on feedback (Prather et al., 2020). Metacognition comprises three components: metacognitive knowledge, metacognitive experiences, and metacognitive strategies (Lee & Mak, 2018). Metacognitive knowledge refers to an individual's knowledge about their own learning, including their cognitive processes related to person, task, and strategy (Flavell, 1979). Metacognitive experiences relate to cognitive or affective experiences that are associated with intellectual tasks. Metacognitive awareness, on the other hand, involves monitoring one's mental state and being aware of the mental processes currently in operation (Loksa, 2020). Metacognitive strategies refer to the ability to utilize strategies to achieve specific goals (Flavell, 1979). Students use these strategies to plan, monitor, and evaluate their own learning (Lee & Mak, 2018). These strategies (Figure 1) include planning, monitoring, and evaluating, which help students control and manage their learning (Pintrich, 1999). Metacognitive strategies have a significant impact on students' academic achievements (Pintrich, 1999).

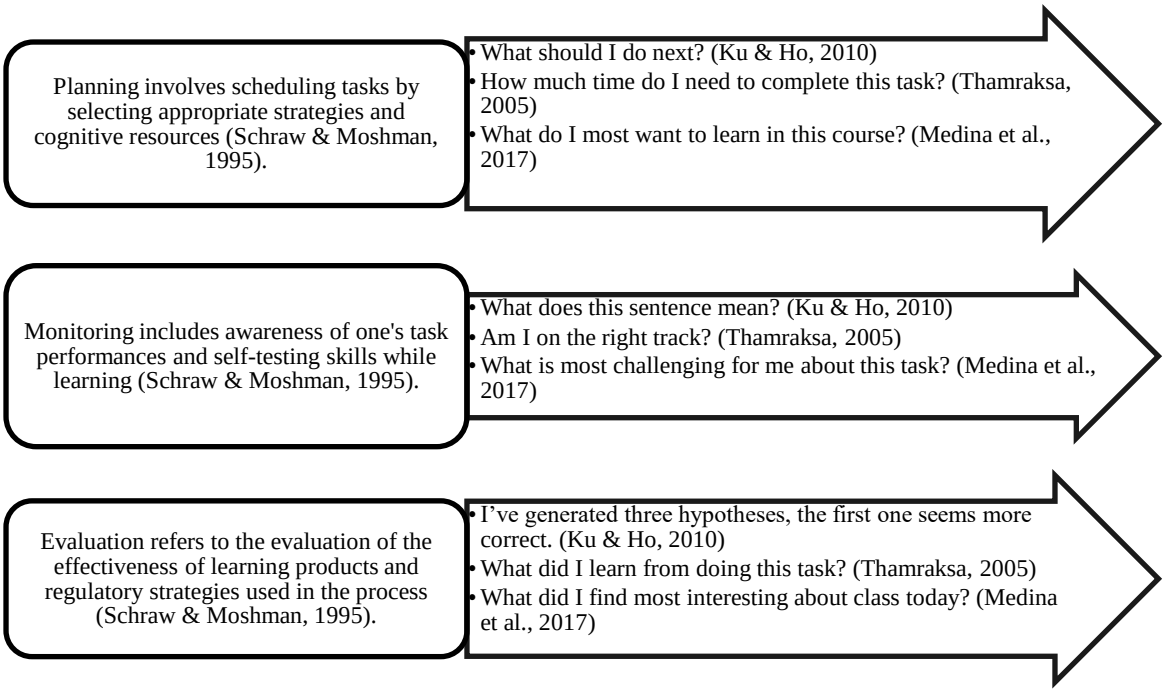

Figure 1. Some examples for planning, monitoring and evaluating strategies

Metacognition in Programming Instruction

Research on metacognition in programming education has gained increasing attention in recent years, with studies revealing the potential for metacognitive skills to aid students in mastering the subject (Prather et al., 2020). Several studies have indicated that programming students struggle with the subject and need to develop cognitive and metacognitive skills to improve their performance (Falkner et al., 2014; Loksa et al., 2022). Reflective activities and structured assignments, such as code explanation and reflective writing, have been reported as effective pedagogical approaches in metacognition studies related to programming education (Prather et al., 2020). Studies have also shown that students who use metacognitive strategies achieve better performance in programming (Bergin et al., 2005) and that interventions designed to enhance metacognitive awareness and problem-solving skills have been effective in improving programming performance (Atmatzidou et al., 2018; Zhou et al., 2021). Enriching programming courses with activities that support metacognition, as well as using structured worksheets, have been found to positively affect the development of metacognitive strategies and improve programming performance (Çakıroğlu & Er, 2020; Yıldız Durak & Uslu, 2022). Additionally, interventions based on metacognition have been found to be more effective in teaching basic programming, and teaching metacognitive strategies has also been shown to improve programming success (Cetin et al., 2014; Wang, 2019). Furthermore, studies have found that metacognitive awareness has a positive effect on basic programming achievement, and programming education can also increase students' metacognitive awareness (Korucu & Atıcı, 2018; Rum & Ismail, 2016).

Online Flipped Classroom

The online flipped classroom is a variation of the traditional flipped classroom, according to Hew et al. (2020). Stöhr et al. (2020) suggest that the online flipped classroom is a promising teaching approach that involves both synchronous and asynchronous online learning. Students are required to watch pre-recorded video lessons with quizzes as preparation for online lessons, instead of in-person lessons in the traditional flipped classroom. Unlike the traditional flipped classroom, the online flipped classroom is conducted remotely with teachers and students connecting online. Since the pandemic, research into the online flipped classroom has increased, with positive outcomes reported in studies such as Gok et al. (2021), Hew et al. (2020), Korkmaz & Mirici (2021), Stöhr et al. (2020), and Tang et al. (2020).

METHOD

In this explanatory sequential under the mixed methods study, the aim is to investigate the impact of metacognitive strategies on the metacognitive awareness of undergraduate students in an online flipped programming course, with a focus on regulatory actions that are relevant to success in programming instruction and distance education learning environments.

The mixed method was preferred because it complements the weak aspects of qualitative and quantitative methods (Creswell & Plano Clark, 2011). This mixed-method study structured in three phases to evaluate the impact of an intervention on participants' metacognitive awareness and strategies. Initially focusing on quantitative measures, the study begins with metacognitive strategy training and assessment through an awareness inventory. During the intervention, qualitative methods such as video materials and metacognitive-oriented forms, along with individual interviews, are employed. After the intervention, the study integrates both

quantitative and qualitative approaches, re-evaluating metacognitive awareness and supplementing with focus group interviews to gather comprehensive insights into the intervention's effectiveness. In current study, the problem was identified and a detailed literature review was carried out. Afterwards, the aim of the study and the sampling method were determined. Data collection tools were determined and developed. The syllabus, content and course materials of the course were prepared. It was decided to give the algorithm design and development course with the online flipped classroom method. Figure 2 and Figure 3 shows the examples of online course and video. Afterwards, the implementation process was carried out (see Fig 4.). This process took 10 weeks.

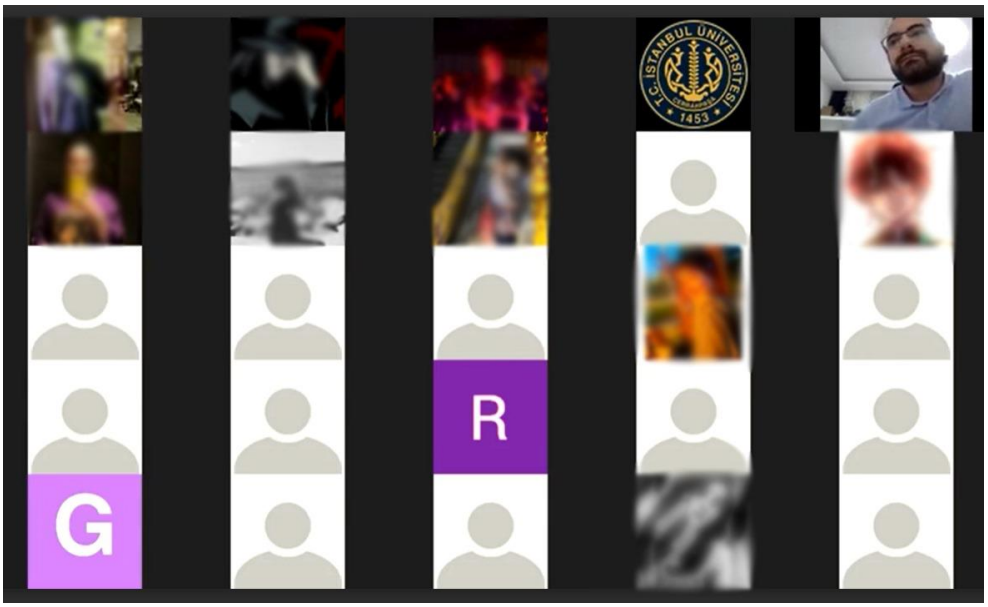

Figure 2. A screenshot from the online courses

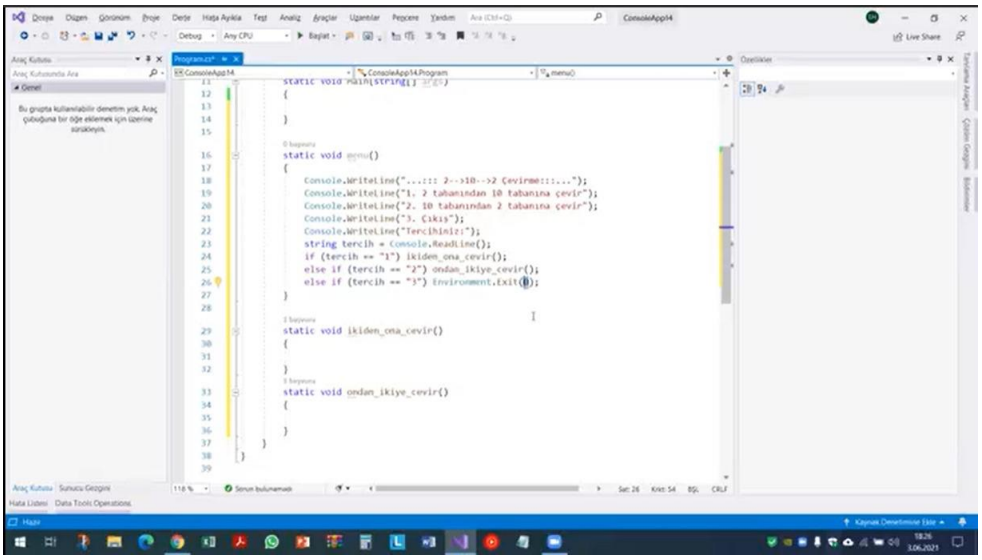

Figure 3. A screenshot from the videos

Participants

Purposive sampling method was used in this study. The participants of this study are 29 freshmen enrolled in the algorithm design and development course. Students are first year students of Computer and Instructional Technologies Department. 30% of the students are males and 70% are females. The average age is 18-20 (%80). 80% of students have not taken a programming course before. 76% of students had never heard of the flipped classroom. "Open-ended questions" and "multiple choice questions" were placed in the videos in order to ensure both interaction and control the viewing status of the students. The number of questions is 3 open ended or 5 close-ended in each video.

Instruments

Metacognitive awareness inventory (MAI), Planning Form, Monitoring Form, Evaluation Form, Semi-Structured Interview Questions for One-on-One Interviews and Semi-Structured Interview Questions for Focus Group Interviews are recruited in this study as instruments. Sample items and questions from instruments are given in Appendix A.

Demographic Form

This form was developed by researchers to obtain demographic information of students. It consists of seven close-ended questions.

Metacognitive Awareness Inventory (MAI)

The MAI, developed by Schraw and Dennison (1994) and adapted into Turkish by Akın et al. (2007), was used in the study. The 5-point Likert-type inventory is a 52-item inventory developed to assess metacognitive awareness. Eight sub-dimensions under the basic dimensions of knowledge and regulation of cognition were obtained. These sub-dimensions are declarative knowledge, procedural knowledge, conditional knowledge, planning, monitoring, evaluation, debugging, and information management. Test-retest reliability coefficients were found to be .95. For our study, the Cronbach's alpha coefficient was found to be .90.

Planning Form

The form was developed by researchers and contains 6 open ended items. Weekly assignment was given to the students. Weekly assignments included an algorithm question (problem). In the first stage of solving the problem, the students filled out the planning form while planning. The form developed to reveal metacognitive schemes of the students while preparing to solve the problem. Each student filled these forms while planning the solution of the problem through Google Forms.

Evaluation Form

It is prepared to reveal the metacognitive schemes after students have solved the algorithm problem. The form consisting of 4 open-ended questions was developed by the researchers. Each student filled these forms when they solved the problem through Google Forms.

Semi-Structured Interview Questions for One-on-One Interviews

These are open-ended questions prepared by researchers to reveal students' metacognitive processes during the process. In the retrospective interviews, questions about feelings (3 questions) under 3 main headings, 3 questions for each of the students who gave and did not give details in the planning/monitoring/evaluation forms, and lastly 2 concluding questions were asked. The form developed by the researchers was examined by an educational technology expert in terms of clarity, comprehensibility and scope. In addition, a pilot study was conducted with 3 students and their opinions were received on the questions.

Semi-Structured Interview Questions for Focus Group Interviews

These open-ended questions prepared by researchers to reveal students' metacognitive processes at the end of the process. These questions were about the planning, monitoring and evaluation processes. Finally, suggestions were asked in the form to organize these processes. These suggestions were classified as form designs, selection expectations, writing habits, expressions for planning-monitoring and evaluation processes. There are 7 open-ended questions in the focus group interview form: questions about feelings, questions about algorithm problems, questions about strategies used and questions about forms. The form developed by the researchers was examined by an educational technology expert in terms of clarity, comprehensibility and scope. In addition, a pilot study was conducted with 3 students and their opinions were received on the questions.

Implementation Process and Procedures

It is possible to consider the stages of the research as before the intervention, during the intervention and after the intervention. In the first week, the students were informed about the purpose of the study, the syllabus, the teaching of the course and the voluntary basis. The study took place over a period of 10 weeks in total (see Appendix B for the course content). The researcher who teaches the course has 10 years of experience in algorithms and programming.

Before the intervention, metacognitive awareness inventory was applied as a pre-test to determine students' metacognitive awareness. In addition, students were given metacognitive strategy lesson, which lasted approximately 1.5 hours. In this way, they have gained knowledge about the basic metacognitive strategies that they can use during this course. Examples of metacognitive strategies are self-questioning, thinking aloud, reflection, and note-taking.

During the intervention, students took the algorithm design and development course in an online flipped classroom environment. In this classroom model, approximately 20-30 minutes of videos prepared by the instructor before the lesson were shown to the students. "Open-ended questions" and "multiple choice questions" were placed in the videos in order to ensure both interaction and control the viewing status of the students. The number of questions is 3 open ended or 5 close-ended in each video.

In the online synchronous lessons, the students' questions were answered after a brief review of the topic. Afterwards, algorithm problems were solved in the practical part. Online synchronous lessons lasted 2-2.5 hours. After each lesson, students were given

an algorithm problem as assignment. Google Classroom was used as the sharing platform of the course and Edpuzzle program was used for sharing course videos. At the end of each lesson, the solutions of the examples covered in the lesson and the lecture notes were shared with the students through Classroom. Students were told that they could ask questions through Classroom and immediate feedback was given by the instructor.

The total number of assignments is six. For the assignment given every week, students were asked to answer in the planning (when planning the solution of the problem), monitoring (while solving the problem) and evaluation (when solving the problem) forms via Google Forms. In addition, students' weekly assignment grades, quiz grades, midterm grades, final grades, weekly video watching grades were recorded during the study period. During the study process, one-on-one interviews were conducted with students which took approximately 30 minutes.

After the intervention, the process was over, MAI was applied to the students as a post-test. In addition, focus group discussions were also conducted. Focus group interviews took approximately one hour. In the one-on-one interviews, three students were selected and interviewed from the students. The number of students randomly assigned to a focus group is in the range of 4-5 people. Audio recording was taken with the permission of the students. The interviews were conducted online. While one of the researchers conducted the one-on-one interviews, both researchers took part in the focus group interviews. Both researchers are experienced in qualitative research methods (see Figure 4.).

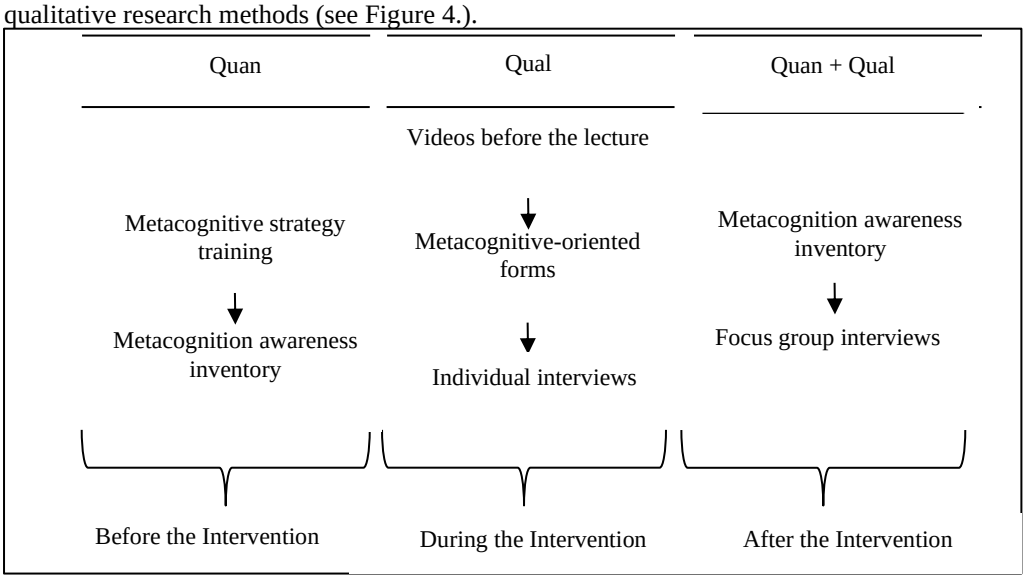

Figure 4. Implementation process of the study

Data Analysis

The data analysis of this study was completed in two stages as the formation of the coding scheme and the analysis of the data of the metacognitive awareness inventory. First, the coding scheme (see Appendix C) was created using the thematic analysis method taking the metacognitive regulation-oriented flipped classroom course design as a central theme (Braun & Clarke, 2006). Second, to compare pre and post-test scores for metacognitive awareness inventory, Wilcoxon Z Test was applied due to the non-parametric nature of the collected data. Replay rates were measured for each student, and it was taken as covariate in Quade nonparametric analysis of covariance.

To note that, the study's reliability and validity are subject to several limitations: the coding scheme's reliability and validity are uncertain due to the low level of inter-rater reliability measures (which was 72%) and rigorous establishment of its alignment with the study's theme. Concerns also arise regarding the reliability of statistical techniques, such as the Wilcoxon Z Test, and the validity of covariate selection, particularly the use of replay rates. These limitations hinder the generalizability of findings, highlighting the need for future research to address these issues through measures like higher inter-rater reliability assessment and validation studies for measurement instruments.

FINDINGS

The first set of research questions aimed to examine the extent to which students' flipped classroom experiences supported their metacognitive processes on the basis of four metacognitive regulatory categories as *orientation, planning, monitoring* and *evaluating.* The first section has analyzed the metacognitive regulations during flipped programming taking metacognitive support forms for programming questions and has argued that *monitoring of progress* and *monitoring of strategy use* were highlighted for designing the course. Whereas the second section questions whether there is a difference between students' metacognitive awareness. In addition, a summary of main findings regarding students' suggestions for their flipped classroom experiences was given.

Metacognitive Regulatory Actions during Flipped Programming Course

Orientation

Orientation for the programming exercises produced four main themes as self-knowledge, other-knowledge, task analysis and content orientation. The most prominent finding for the orientation category was finding *task connections* (*f*=22) before the students were planning for the solution. Moreover, they mentioned about their awareness of own feelings that they have felt before planning the solution as challenging, enjoyment, achievement, stressed, uneasiness, or curiosity. Students mentioned about the negative feelings as challenging, stressed or uneasiness at the beginning that they first encountered with the question. Otherwise, the positive feelings as achievement and enjoyment were experienced by students at the end of their solution. Students mainly produced utterances about their peers' comprehensibility for a variety of conditions when their solutions can be predictable, declarative, having unique algorithm design and connected to lesson. Also, they mentioned about filling the form approach as they are oriented to be aware of different strategies during problem solving, and to be aware of the steps they will go through to get along with the solution (see Table 1).

Content orientation before they plan the problem solution was another important issue for the students. They mentioned when they first encounter a programming question, they perform the processes for activating prior knowledge, anticipating the potential solution and anticipating new knowledge at the end of the solution. Following utterance is an example of activating prior knowledge for content orientation to programming exercise:

"First of all, I was checking my information. Do you have this information? I was starting the question accordingly."

Table 1. Students' utterances regarding orientation to the programming exercise

		First	Second	Total
Self-knowledge				
	Alternative solution ways	6	1	7
	Knowledgeability	6	3	9
	Study habits	7	4	11
	Feelings	21	17	38
Other-knowledge				
	Filling the form approach	12	12	24
	Knowledgeability	2	0	2
	Comprehensibility	9	0	9
	Instructor's approach	1	0	1
Task analysis				
	Task materials	2	1	3
	Task connections	16	6	22
	Task difficulty	0	2	2
	Task demands	2	1	3
Content orientation				
	Anticipating solution	5	1	6
	Anticipating new knowledge	4	1	5
	Activating prior knowledge	8	1	9

Planning

Planning during the programming produced two main themes as planning in advance, and interim planning (see Table 2). Both of them provided utterances in regard of task objectives and task solution approach. During the planning phase, students mentioned they had a tendency to identify the approach for solving the task. Considering the shortest route and transforming flowchart to code blocks were embraced in planning in advance by the students; whereas, self-questioning and considering the shortest route were considered for interim planning. For the intersection of planning action for both interviews and for both sub-themes of planning, considering the shortest route was uttered frequently compared to other sub-actions by the students. Following utterance was for task solution approach during interim planning:

"Sometimes while I was designing, when I used decision structures, I wondered if it was tiring. As mentioned in the question, I thought there might be differences in these aspects to see if there could be a shorter path."

Table 2. Students' utterances regarding planning the programming exercise

		First	Second	Total
Interim planning				
	Task objectives	5	0	5
	Task solution approach	5	1	6
Planning in advance				
	Task objectives	1	0	1
	Task solution approach	8	0	8

Monitoring

Monitoring revealed three main themes as comprehension monitoring, monitoring of progress and monitoring of strategy use. While the majority of the students had utterances focused on error detection in the middle of the semester, they frequently stated that they showed error correction behavior in the interviews held at the end of the semester (see Table 3). Students mentioned that they demonstrated error correction and error detection behaviors during their programming exercise.

"Write characters properly. Using neat shapes. Then, when trying at the exit part. Does it give correct answers at all values? Like we found the truth by mistake. I was generally reviewing, but I was also hitting on the character writing."

Similarly, the majority of the students produced more discourse on using their individual strategical preferences and strategies used in the course during solving programming exercise.

Table 3. Students' utterances regarding monitoring during solving the programming exercise

		First	Second	Total
Comprehension monitoring				
	Demonstrating comprehension by differentiating	7	4	11
	Noticing lack of comprehension	5	0	5
Monitoring of progress				
	Error correction	14	17	31
	Error detection	19	5	24
Monitoring of strategy use				
	Feeling-based	3	0	3
	Person-based	15	34	49
	Course-based	13	19	32
	Example-based	2	8	10

Evaluating

Evaluating revealed a total of six subthemes under evaluating learning process and evaluating learning outcome's main themes (see Table 4). Reflecting on task demand and self-efficacy revealed the highest proportion of the students' utterances. Students mentioned the difference between open-ended and multiple-choice questions and reflected their needs to take feedbacks on their answers to the questions to be sure as a facilitator to lower the task demand. Following utterance is an example of reflecting on task demand:

"I used to get stuck with open-ended ones, usually what kind of answer should be given. I wasn't sure how to respond. Since I couldn't predict those open-ended multiple-choice, multiple-choice seemed better to me."

Table 4. Students' utterances regarding evaluating the programming exercise

		First	Second	Total
Evaluating learning process				
	Reflecting on task demand	2	18	20
	Reflecting on task difficulty	0	1	1
	Reflecting on self-efficacy	12	4	16
Evaluating learning outcomes				
	Checking clarity of the solution	3	0	3
	Reflects on effectiveness of the solution	4	1	5
	Checking correctness of the solution	4	1	5

Metacognitive Awareness Inventory

Metacognitive awareness inventory contains six factors to compare for pre and post-test scores. Although it does not reveal any significant difference for pre and post test scores when Wilcoxon Z Test was applied, it seems that monitoring and planning produced a differentiation for the mean scores. While mean value of monitoring was 3.72 for the pre-score, it was 3.39 for the post-score (see Table 5).

Table 5. Non-Parametric Wilcoxon Z Test for Pre and Post MAI Scores

Scale Factor	Form period (n=23)				Z Wilcoxon	p
	Pre		Post			
	Mean	Std. Dev.	Mean	Std. Dev.		
Declarative knowledge	.53	.15	.57	.08	1.14	.25
Procedural knowledge	.31	.12	.27	.05	.85	.39
Conditional knowledge	.35	.06	.36	.04	.52	.59
Planning	.48	.10	.45	.10	1.15	.24
Monitoring	.56	.13	.51	.10	-1.84	.06
Evaluating	.40	.09	.40	.08	.07	.94
Debugging	.34	.08	.35	.05	.43	.63
Information Management	.64	.10	.62	.09	-.84	.39
Total awareness score	3.64	.68	3.58	.44	-.62	.53

Replay Rate as Covariate in Quade Nonparametric Analysis of Covariance

When replay rate is taken as covariate for the pre and post test scores, mean differences between the declarative, procedural, condition knowledge, monitoring, evaluating, debugging, and information management showed significance. However, planning does not result in any significant difference for the replay rate as a covariate. The total value showed a significant difference as well: $T(44)=6.120$, $p<0.01$

Students' Overall Suggestions for the Programming Course-Design

On Filling the Form Approach: The evaluations made by the students regarding the form filling approach were evaluated under the "other-knowledge as an orientation issue" in general terms. At the end of the study, the students expressed their suggestions about this approach, including triggering the planned activity, increasing awareness of different learning strategies, and considering that writing takes time. They also mentioned the need for hint and feedback availability for filling the question form. The following discourses include the evaluations of the students about this approach:

"My teacher, I felt the need to check myself again. While doing that homework, I say to myself again like the question on the form, don't be distracted, go to a quiet place, put the phone down. Because of that question on the form, I always have to focus like this. This is my favorite part. I'm giving myself self-control." (Triggering planned activity)

"In other words, if there is such a convenience, I wonder if they think something wrong in the form, while designing the algorithm, you can know from there whether they are progressing correctly, such small feedbacks can be good." (Need for feedback availability)

"For example, from my point of view, when I first started, for example, I did not understand what to do. You know, little hints there, hints about what you want to do here would be nice." (Need for hint availability)
"As a strategy, I started to learn a little more about what strategy is, thanks to the forms you provided." (Awareness of different strategies)

"It seemed a bit long to write the sequence I envisioned here." (Take long time to write)

On Open Ended and Multiple-Choice Questions Embedded in Videos: Within the course videos, two types of questions, open-ended and multiple-choice, were presented to the students. Due to the need for feedback, most of the students suggested that multiple choice questions should be increased or feedback should be provided in open-ended questions:

"I don't know if I made the right or wrong answers to the classical questions. I shoot and that question reverberates in my head that week. I wonder what will happen if I did wrong, should I go back? That's why I don't like classical questions at all." (Feedback availability)

"I used to get stuck with open-ended ones, usually what kind of answer should be given. I wasn't sure how to respond. Since I couldn't predict those open-ended multiple-choice, multiple-choice seemed better to me." (Feedback availability)

"Do we understand the video at once and it shows it." (Video comprehension)

On Video-Represented Course: The majority of the students stated that presenting videos in the programming course facilitated the learning process. Students suggested that it is beneficial for them to watch the video repeatedly during their comprehension process, so the presentation of the videos should continue.

"I also need to watch the previous lesson video before I start, otherwise I can't do it. I'm trying to do it by looking at the algorithm that the teacher asked, which one did it with." (Simultaneous programming)

CONCLUSION, DISCUSSION AND SUGGESTIONS

This study aimed to examine the metacognitive regulatory actions of undergraduate students and the impact of a form approach on their metacognitive awareness in the context of a flipped programming course. This study provides a coding scheme for programming education and found that, with replay rate as a covariate, metacognitive awareness scores were significant except for the planning process. The results suggest that the processes frequently used by undergraduates might reveal their needs for a basic programming course.

The findings derived from the orientation segment of programming exercises reveal several significant insights into how students approach problem-solving in programming tasks. The four key themes - *self-awareness, external knowledge, task analysis, and content comprehension* - illuminate various aspects of students' cognitive and emotional engagement in the problem-solving process. The most noteworthy discovery, highlighted 22 times, is the prioritization of identifying task connections prior to formulating a solution. This indicates that students attribute importance to comprehending the context and requirements of the task, suggesting a strategic problem-solving approach that emphasizes thorough analysis and comprehension before initiating solution planning. This underscores the crucial role of task analysis in programming education, serving as a cornerstone for crafting effective problem-solving methodologies. Çakıroğlu and Er (2020) echoed a similar perspective in the context of flipped programming education. Hence, the necessity for students to seek guidance and establish task connections before tackling programming problems may be significant. Consequently, there could be a requirement for a supportive system that offers examples enabling novice students to establish task connections before commencing programming exercises, non-example solutions, or process breakdowns to facilitate their ability to do so. In this scenario, the questions provided in the forms for students' planning processes could be construed as guiding tasks. Additionally, besides endorsing this proposal, integrating a feature into the system allowing students to receive hints and automated feedback for refining the suggestion could be contemplated for future research.

Moreover, an intriguing discovery was the observation of undergraduates frequently mentioning the shortest route while planning compared to other sub-actions. This might be due to the form approach promoting a common language among students during programming exercises. This supports the idea that planning involves forethought and activation of prior knowledge (Loksa et al., 2022). Additionally, the rare mention of planning processes in the second set of interviews suggests automation of this process as students practice. This aligns with previous research by Lui et al. (2006) which found that novice programmers improve and become more efficient as they gain experience.

A significant finding was that the Quade nonparametric analysis of covariance revealed a difference in undergraduates' metacognitive awareness scores pre and post-test except for the planning process, when replay rate was used as the covariate. Although the frequency of planning process utterances decreased between the two sets of interviews, the lack of significance even with replay rate as the covariate may require further investigation. It may be that planning is automated as students gain proficiency in regulating their metacognitive actions, and thus awareness of planning is not impacted by differences in replay rates. This could be due to students' preference for replaying after monitoring and evaluating phases during programming exercises. Furthermore, it is hypothesized that students might prefer to replay after the monitoring and evaluating phases during programming exercises. This preference might be influenced by their cognitive strategies (Ahsan & Obeidellah, 2021), or prior experiences (Loksa & Ko, 2016), which could impact how they engage with the replay function and subsequently affect their metacognitive awareness during different phases of problem-solving in a programming task. In conclusion, contrary to the findings of the study, an enhancement in students' metacognitive awareness as a result of the flipped classroom approach (Limueco & Prudente, 2019; van Alten et al., 2020) was not observed when the covariance of replay rate was not taken into account.

In addition, the data analysis revealed that in the second interview, the students reported a higher frequency of statements related to error correction than error detection. It's possible that this shift in perception is due to the students' understanding of error detection and correction as interrelated processes by the end of the course. However, it's unclear whether the method of form approach or video learning influenced this perception. Furthermore, the specific type of errors that the students were referring to in this qualitative finding cannot be determined. While previous studies have suggested that syntactic errors have a positive effect on programming learning (Beege et al., 2021), it's uncertain which aspect of the flipped course design may have influenced the reported outcome. Therefore, several uncertainties necessitate further exploration. Firstly, it remains unclear whether the instructional methods utilized in the course, such as the form approach or video learning, influenced the observed change in students' perceptions regarding error detection and correction. Understanding how different teaching approaches impact students' learning processes is essential for crafting effective instructional strategies in programming education. Moreover, the specific categories of errors discussed by

students are unspecified, which limits the depth of analysis. Diverse types of errors may exert distinct influences on students' learning journeys and achievements. For instance, while prior research suggests that syntactic errors may enhance programming learning, it remains uncertain whether this factor contributed to the observed shift in student perspectives.

It is noteworthy that students appeared to contemplate their self-efficacy while completing programming exercises in the middle of the semester, but shifted their focus towards the task demands of the exercises towards the end of the semester. The change in reflection patterns observed among students, from self-efficacy during the middle of the semester to task demand at the end of the semester, could potentially be attributed to increased academic demands and exam pressures towards the end of the term, rather than solely due to the programming course. This emotional strain may lead to a decline in students' self-efficacy. While previous studies have not encountered findings supporting this notion, evidence has been found indicating an increase in self-efficacy in middle school student groups participating in robot programming activities (Yıldız Durak et al., 2019). Similarly, in adult groups, software development has been associated with a positive attitude towards programming and an increasing perception of self-efficacy (Kovari & Katona, 2023). Although no experimental research supporting this finding has been identified at this stage, future research addressing this gap could be crucial.

The results of this study are significant in that they shed light on the positive influence of metacognitive regulation support on students' metacognitive awareness and programming learning. Although the findings require further examination, they offer valuable insights for the design of flipped programming courses. It's important to note that the students' self-efficacy perceptions may decline towards the end of the term, and therefore, self-efficacy could also be measured alongside metacognitive awareness in future studies. Overall, this study provides a comprehensive understanding of flipped course design for programming instruction and points to the need for further research in this area.

Disclosure Statement
There is no conflict of interest for this study.

Ethics and Consent: Ethics committee approval for this study was received from the Ethics Committee of Manisa Celal Bayar University (Date: 12.02.2021; Approval Number: 2021/03, E--050.01.04-26870)

REFERENCES

Ahsan, Z., & Obaidellah, U. (2021). Visual behavior on problem comprehension among novice programmers with prior knowledge. *Procedia Computer Science,* 192, 2347–2354. https://doi.org/10.1016/j.procs.2021.09.003

Akın, A., Abaci, R., & Cetin, B. (2007). Bilişötesi Farkındalık Envanteri'nin Türkçe Formunun Geçerlik ve Güvenirlik Çalışması. *Educational Psychology*, 67, 483-496.

Anderson, N. J. (2002*). The Role of Metacognition in Second Language Teaching and Learning.* ERIC Digest.

Atmatzidou, S., Demetriadis, S., & Nika, P. (2018). How does the degree of guidance support students' metacognitive and problem solving skills in educational robotics?. *Journal of Science Education and Technology*, *27*, 70-85.

Avcı, Ü. (2022). A predictive analysis of learning motivation and reflective thinking skills on computer programming achievement. *Computer Applications in Engineering Education*, *30*(4), 1102-1116.

Beege, M., Schneider, S., Nebel, S., Zimm, J., Windisch, S., & Rey, G. D. (2021). Learning programming from erroneous worked-examples. Which type of error is beneficial for learning? *Learning and Instruction*, *75*, 101497. https://doi.org/10.1016/J.LEARNINSTRUC.2021.101497

Bergin, S., Reilly, R., & Traynor, D. (2005). Examining the role of self-regulated learning on introductory programming performance. In *Proceedings of the first international workshop on Computing education research* (pp. 81-86).

Bernard, M., & Bachu, E. (2015). Enhancing the metacognitive skill of novice programmers through collaborative learning. *Metacognition: Fundaments, Applications, and Trends: A Profile of the Current State-Of-The-Art*, 277-298.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*(2), 77–101. https://doi.org/10.1191/1478088706qp063oa

Broadbent, J., & Poon, W. L. (2015). Self-regulated learning strategies & academic achievement in online higher education learning environments: A systematic review. *The internet and higher education*, *27*, 1-13.

Çakıroğlu, Ü., & Er, B. (2020). Effect of using metacognitive strategies to enhance programming performances. *Informatics in Education*, *19*(2), 181-200.

Cetin, I., Sendurur, E., & Sendurur, P. (2014). Assessing the impact of meta-cognitive training on students' understanding of introductory programming concepts. *Journal of Educational Computing Research*, *50*(4), 507-524.

Clark, R. M., Kaw, A. K., & Braga Gomes, R. (2022). Adaptive learning: Helpful to the flipped classroom in the online environment of COVID?. *Computer Applications in Engineering Education*, *30*(2), 517-531.

Creswell, J. W. & Plano Clark, V. L. (2011). Designing and conducting mixed methods research. Los Angeles, CA: Sage.

Falkner, K., Vivian, R., & Falkner, N. J. (2014). Identifying computer science self-regulated learning strategies. In *Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 291-296).

Flavell, J. H. (1979). Metacognition and cognitive monitoring—A new era of cognitive-developmental inquiry. *American Psychologist,* 34, 906–911.

Gok, D., Bozoglan, H., & Bozoglan, B. (2021). Effects of online flipped classroom on foreign language classroom anxiety and reading anxiety. *Computer Assisted Language Learning*, 1-21.

Hew, K. F., Jia, C., Gonda, D. E., & Bai, S. (2020). Transitioning to the "new normal" of learning in unpredictable times: pedagogical practices and learning performance in fully online flipped classrooms. *International Journal of Educational Technology in Higher Education*, *17*, 1-22.

Jia, C., Hew, K. F., Jiahui, D., & Liuyufeng, L. (2023). Towards a fully online flipped classroom model to support student learning outcomes and engagement: A 2-year design-based study. *The Internet and Higher Education*, *56*, 100878.

Karatas, K., & Arpaci, I. (2021). The role of self-directed learning, metacognition, and 21st century skills predicting the readiness for online learning. *Contemporary Educational Technology*, *13*(3).

Korkmaz, S., & Mirici, İ. H. (2021). Converting a conventional flipped class into a synchronous online flipped class during COVID-19: university students' self-regulation skills and anxiety. *Interactive Learning Environments*, 1-13.

Korucu, A. T., & Atıcı, K. (2018). The determination of metacognitive awareness situations of secondary school students receiving programming education with Alice. *Journal of Learning and Teaching in Digital Age*, *3*(1), 3-11.

Kovari, A., & Katona, J. (2023). Effect of software development course on programming self-efficacy. *Education and Information Technologies*, 28(9), 10937–10963. https://doi.org/10.1007/s10639-023-11617-8

Ku, K. Y., & Ho, I. T. (2010). Metacognitive strategies that enhance critical

Lee, I., & Mak, P. (2018). Metacognition and metacognitive instruction in second language writing classrooms. *tesol QUARTERLY*, *52*(4), 1085-1097.

Limueco, J. M., & Prudente, M. S. (2019). Flipped classroom enhances student's metacognitive awareness. *ACM International Conference Proceeding Series*, 70–74. https://doi.org/10.1145/3306500.3306507

Loksa, D. (2020). *Explicitly Training Metacognition and Self-Regulation for Computer Programming*. University of Washington.

Loksa, D., & Ko, A. J. (2016). The Role of Self-Regulation in Programming Problem Solving Process and Success. *Proceedings of the 2016 ACM Conference on International Computing Education Research*. https://doi.org/10.1145/2960310.2960334

Loksa, D., Ko, A. J., Jernigan, W., Oleson, A., Mendez, C. J., & Burnett, M. M. (2016). Programming, problem solving, and self-awareness: Effects of explicit guidance. In *Proceedings of the 2016 CHI conference on human factors in computing systems* (pp. 1449-1461).

Loksa, D., Margulieux, L., Becker, B. A., Craig, M., Denny, P., Pettit, R., & Prather, J. (2022). Metacognition and self-regulation in programming education: Theories and exemplars of use. *ACM Transactions on Computing Education (TOCE)*, *22*(4), 1-31.

Lui, K. M., & Chan, K. C. C. (2006). Pair programming productivity: Novice–novice vs. expert–expert. *International Journal of Human-Computer Studies*, *64*(9), 915–925. https://doi.org/10.1016/J.IJHCS.2006.04.010

Mayer, R. E. (1998). Cognitive, metacognitive, and motivational aspects of problem solving. *Instructional science*, *26*(1-2), 49-63.

Medina, M. S., Castleberry, A. N., & Persky, A. M. (2017). Strategies for improving learner metacognition in health professional education. *American journal of pharmaceutical education*, *81*(4).

Nurulain Mohd Rum, S., & Zolkepli, M. (2018). Metacognitive strategies in teaching and learning computer programming. *International Journal of Engineering & Technology, 7*, 788–794.

Pea, R. D., Soloway, E., & Spohrer, J. C. (1987). The buggy path to the development of programming expertise. *Focus on Learning Problems in Mathematics*, *9*, 5-30.

Pintrich, P. R. (1999). The role of motivation in promoting and sustaining self-regulated learning. *International journal of educational research*, *31*(6), 459-470.

Polat, E., Hopcan, S., & Arslantaş, T. K. (2022). The association between flipped learning readiness, engagement, social anxiety, and achievement in online flipped classrooms: a structural equational modeling. *Education and Information Technologies*, *27*(8), 11781-11806.

Prather, J., Becker, B. A., Craig, M., Denny, P., Loksa, D., & Margulieux, L. (2020, August). What do we think we think we are doing? Metacognition and self-regulation in programming. In *Proceedings of the 2020 ACM conference on international computing education research (pp. 2-13).*

Rum, S. N., & Ismail, M. (2016). Metacognitive awareness assessment and introductory computer programming course achievement at university. *Int. Arab J. Inf. Technol.(IAJIT)*, *13*, 667-675.

Scherer, R., Siddiq, F., & Viveros, B. S. (2020). A meta-analysis of teaching and learning computer programming: Effective instructional approaches and conditions. *Computers in Human Behavior*, *109*, 106349.

Schoenfeld, A. H. (2016). Learning to think mathematically: Problem solving, metacognition, and sense making in mathematics (Reprint). *Journal of education*, *196*(2), 1-38.

Schraw, G. & Dennison, R. S. (1994). Assessing metacognitive awareness. Contemporary Educational Psychology, 19, 460-475.

Schraw, G., & Moshman, D. (1995). Metacognitive theories. *Educational psychology review*, *7*, 351-371.

Stöhr, C., Demazière, C., & Adawi, T. (2020). The polarizing effect of the online flipped classroom. *Computers & Education*, *147*, 103789.

Sun, L., Guo, Z., & Zhou, D. (2022). Developing K-12 students' programming ability: A systematic literature review. *Education and Information Technologies*, *27*(5), 7059-7097.

Tang, T., Abuhmaid, A. M., Olaimat, M., Oudat, D. M., Aldhaeebi, M., & Bamanger, E. (2020). Efficiency of flipped classroom with online-based teaching under COVID-19. *Interactive Learning Environments*, 1-12.

Thamraksa, C. (2005). Metacognition: A key to success for EFL learners. *BU Academic Review*, *4*(1), 95-99.

Tsai, C. W., Lee, L. Y., Cheng, Y. P., Lin, C. H., Hung, M. L., & Lin, J. W. (2022). Integrating online meta-cognitive learning strategy and team regulation to develop students' programming skills, academic motivation, and refusal self-efficacy of Internet use in a cloud classroom. *Universal Access in the Information Society*, 1-16.

van Alten, D. C. D., Phielix, C., Janssen, J., & Kester, L. (2020). Self-regulated learning support in flipped learning videos enhances learning outcomes. *Computers and Education*, *158*. https://doi.org/10.1016/J.COMPEDU.2020.104000

Wang, Y. (2019). Study of Metacognitive Strategies' Impacts on C Language Programming Instruction. In *2nd International Conference on Contemporary Education, Social Sciences and Ecological Studies (CESSES 2019)* (pp. 112-116). Atlantis Press.

Yıldız Durak, H., Karaoğlan Yilmaz, F. G.., & Yilmaz, R. (2019). Computational Thinking, Programming Self-Efficacy, Problem Solving and Experiences in the Programming Process Conducted with Robotic Activities. *Contemporary Educational Technology*, 10(2), 173–197. https://doi.org/10.30935/cet.554493

Yıldız Durak, H., & Atman Uslu, N. (2022). Investigating the effects of SOLO taxonomy with reflective practice on university students' meta-cognitive strategies, problem-solving, cognitive flexibility, spatial anxiety: an embedded mixed-method study on 3D game development. *Interactive Learning Environments*, 1-23.

Yilmaz, R. M., & Baydas, O. (2017). An examination of undergraduates' metacognitive strategies in pre-class asynchronous activity in a flipped classroom. *Educational Technology Research and Development*, *65*, 1547–1567. https://doi.org/10.1007/s11423-017-9534-1

Yılmaz, R., & Keser, H. (2017). The impact of interactive environment and metacognitive support on academic achievement and transactional distance in online learning. *Journal of Educational Computing Research*, *55*(1), 95-122.

Zhao, L., & Ye, C. (2020). Time and performance in online learning: Applying the theoretical perspective of metacognition. *Decision Sciences Journal of Innovative Education*, *18*(3), 435-455.

Zhou, P., Li, J., Chen, F., Zhou, H., Bao, S., & Li, M. (2021). Design of metacognitive scaffolding for k-12 programming education and its effects on students' problem solving ability and metacognition. In 2021 Tenth International Conference of Educational Innovation through Technology (EITT) (pp. 182–186). IEEE.

APPENDICES

Appendix A. Data Collection Tools

Instruments	Reference	Sample Items/Questions
Metacognitive awareness inventory (MAI)	Akın et al. (2007)	• I consider a few alternatives before answering a problem. • I try to use the strategies I have used in the past.
Demographic Form	Developed by researchers	• Age • Gender
Planning Form	Developed by researchers	• Deciding on my prior knowledge about the algorithm I will design • Before I start my algorithm design, asking myself questions.
Monitoring Form	Developed by researchers	• Questioning whether strategies different from those I have applied can be applied to complete the algorithm design • Checking if I am distracted while completing the algorithm design
Evaluation Form	Developed by researchers	• Deciding whether I have achieved the task given as a result of my algorithm design • Summarizing what I learned after algorithm design
Semi-Structured Interview Questions for One-on-One Interviews	Developed by researchers	1. How did you feel while answering the planning questions? 2. In the planning forms, it seems that you wrote short answers such as "I tried to give, I have enough information". Is there any particular reason for this? a. What did you pay attention to before creating the algorithm? b. What did you include in your planning? c. Why did you decide to include them in your planning?
Semi-Structured Interview Questions for Focus Group Interviews	Developed by researchers	What's your first step in an algorithm question? What was your favorite aspect of form designs? Why?

Appendix B. Course Content

Algorithm 1st Week Topics	Learning Objectives
1. What is an algorithm	- To understand what an algorithm is and its purpose.
2. Flow charts and figures	- To gain the ability to represent algorithms visually using flowcharts and figures.
3. Algorithm printing	- To develop the ability to write algorithms in a structured format.
Algorithm 2nd Week Topics	
1. Using FcPro3	- To gain the ability to use FcPro3 flowcharting program.
2. Definition of operators (+,-,/,*,%)	- To define mathematical operators (+, -, /, *, %) and apply them in algorithm design.

3. Condition structures	- To use condition structures to direct the flow of the algorithm.
4. Logical operators (==, <, >, <=, >=, !=, <>, !<, !>, &&, ||)	- To understand and use logical operators (==, <, >, <=, >=, !=, <>, !<, !>, &&, ||) for creating complex conditions.
Algorithm 3rd week Topics	
1. Loops	To master the concept and use of loops to perform repetitive tasks within an algorithm.
Algorithm 4th weekTopics (C# Lesson 1)	
1. Using Visual Studio	- To learn for installing Visual Studio and use it for C# programming.
2. Printing code on the screen	- To do code printing on the screen with C#.
3. Using arithmetic operations with C#	- To apply arithmetic operations in C# applications.
4. What is the description line, how to create it	- To understand what comment lines are and learn how to create them.
5. What are bool, string, integer, char, double and where are they used	- To learn the use of basic data types such as bool, string, integer, char, double and the differences between them.
Algorithm 5th week Topics (C# Lesson 2)	
1. Receiving data from user	- To gain the ability to receive data from the user and process this data in a C# program.
2. If – else, else if	- To apply conditional constructs such as if-else, else if to control program flow.
3. Switch-case	- To understand and use switch-case constructs for multiple branch options.
4. Use of codes like ToLower – ToUpper	- To use methods such as ToLower and ToUpper to convert string expressions to upper/lower case.
Algorithm 6th week Topics (C# Lecture 3)	
1. Nested conditional structures	- To understand and create nested conditional structures for complex decision making.
Algorithm 7th weekTopics (C# Lecture 4)	
1. While	- To use the while loop for iterative execution based on a conditional state.
2. For	- To learn the for loop to iterate over a given number of steps.
3. Do while	- To understand the do while loop and its unique execution flow for repeated operations.
4. Random use	- To understand random number generation in C# and its applications.
Algorithm 8th weekTopics (C# Lecture 5)	
1. Use of nested loops	To master the use of nested loops to handle multi-dimensional operations.
a. An example of nested for loop structure is given in the video	
Algorithm 9th week Topics (C# Lecture 6)	
1. What are arrays, where do we use them	- To understand what arrays are and where they are used.
2. Index definition	- To understand what indexing is and how to access elements in arrays.
3. Creating an array and assigning values	- To create arrays in C# and assigning values to them.
Algorithm 10th week Topics (C# Lecture 7)	
1. Break – Continue	- To use break and continue statements to control loop execution.
2. Return	- To use return statements to return values from functions.
3. Use of functions and their intended use	- To understand the creation and use of functions to modularize code.
4. What are the concepts of receiver and giver	- To understand the concepts of 'receiver' and 'transmitter' in the context of function parameters and return values.

Appendix C. Coding Scheme for Interviews

Category	Theme	Sub-theme	Definition	Example Utterance
Orientation	Content	Activating prior knowledge	Prior knowledge regarding the content including operators, course experience, other-experience (which may include) is being activated at the first sight of the algorithm question.	Eng: "First of all, I was checking my information. Do you have this information? I was starting the question accordingly." Tr: "öncelikle bilgilerimi kontrol ediyordum. bu bilgiler var mı yok mu. ona göre soruya başlıyordum."
		Anticipating solution		Eng: "I think in my head, if I design it in an algorithm, I will get results." Tr: "Kafamda düşünüyorum bunu nasıl bir algoritmada tasarlarsam sonuç elde ederim diye."
		Anticipating new knowledge	At the end of the algorithm, one anticipates which kind of operations regarding the algorithm design	Eng: "I need information first. I think it is necessary to know the meanings of those decision structures and the structures used in the flowchart. that information is needed. If we are going to do a homework about numbers, which is also necessary for homework, there is a need to know the meanings of those numbers."

Category	Theme	Sub-theme	Definition	Example Utterance
			will be acquired as new knowledge.	Tr: "önce bilgiye ihtiyacım. o karar yapıların, akış şemasında kullanılan yapıların anlamlarını bilmek gerektiğini düşünüyorum ki do yapısını ben aslında bu dersin ödevlerini yaparken daha iyi kavradım. o bilgiye ihtiyaç var. aynı zamanda ödev için gerekli olan, sayılar ile ilgili bir ödev yapacaksak o sayıların anlamlarını bilmeye ihtiyaç var."
	Task analysis	Task demands	One specifies task demands as how much time may it take, what kind of operators may be used, how may the question is different from the previous ones regarding its demands.	Eng: "I have to determine which operation you want. Which way should I go? detail of the operation to be performed, will the counter or the loop be used? I tried to look at these kinds of things" Tr: "hangi işlemi istediğini onu belirlemek zorundayım. hangi yoldan gitmem gerektiğini. yapılacak işlemin ayrıntısını, sayaç mı kullanılacak döngü mü kullanılacak? bu tarz şeylere bakmaya çalıştım"
		Task difficulty	One specifies the question difficulty regarding the demand, differences, time and familiarity.	Eng: "The first thing I did after I realized how hard it was for me" Tr: "anladıktan sonra yaptığım ilk şeyde benim için ne kadar zor olup olmadığı"
		Task connections	One specifies the connections within the question and between the question. As semestre continues, these connections are getting complex.	Eng: "It requires more than one condition. It was a little more difficult to figure out where to reach which result. When I didn't take a step, the conditions were mixed." Tr: "birden fazla koşul istiyor. hangi sonuca nereden ulaşacağımı bulmak biraz daha zor oluyordu. adım atmayınca koşullar birbirine giriyordu."
		Task materials	One specifies the materials needed for the task completion.	Eng: "If it is an algorithm that will force me, I take paper and a pen and write" Tr: "eğer beni zorlayacak bir algoritma ise elime kâğıt, kalem alıp yazıyorum"
	Self-knowledge	Awareness of own feelings	One is aware of own feelings regarding the algorithm question and form.	Eng: "When we didn't do an assignment, there was a task phase, a plan phase, and I was writing them down on forms. In a way, this is the planning phase of the assignment, so I enjoyed it while filling it out. At first, yes, it was a bit different, boring, but then I enjoyed it." Tr: "bir ödevi yapmadığımızda bir görev aşaması, plan aşaması oluyordu ve bunları formlara yazıyordum. Bu bir bakıma ödevi planlama aşaması olduğu için ben zevk aldım yani doldururken. İlk başta evet biraz değişik sıkıcı geldi ama sonradan keyif aldım."
		Awareness of alternative solution ways		Eng: "Because, my teacher, shorter structures are said in the lesson, teacher, if we do this, it will be shorter. There are things that are used interchangeably, I believe, there are definitely different solutions." Tr: "Çünkü hocam derste de daha kısa yapılar söyleniyor hocam bunu yaparsak daha kısa olur. birbirinin yerine kullanılan şeylerde var ben inanıyorum mutlaka farklı çözüm yolları da vardır."
		Self-knowledge of study habits	One has self-knowledge on how one may prefer to study for applied courses.	Eng: "I don't like to mess with a strategy or something I'm doing. For example, if I am rote in biology, I will continue to rote. You know, when I turn it into something else, it looks like it will break, but the teacher had already told us in only one of my assignments. It was on my mind. When I did the coding directly without drawing anything, it worked in the program. I'm firm on some things. I don't spoil anything much. So I don't know if it's regular or not. In other words, if I can continue with something that is complete, I am not in favor of changing it much." Tr: "Bir stratejim ya da yaptığım bir şeyi çok bozmayı sevmem. Mesela biyolojide ezbere gidiyorsam ezbere gitmeye devam ederim. Hani onu başka bir şeye çevirdiğimde sanki bozulacak gibisinden ama sadece bir tane ödevimde hoca zaten bize anlatmıştı. O aklımda kalmıştı. Hiç çizmeden bir şey yapmadan kodlamayı direk yaptığımda programda çalıştı. Bazı şeylerde sabit fikirliyimdir. Çok bir şeyleri bozmam. Yani düzenli mi oluyor onu da tam bilmiyorum. Yani yaptığım bir şeyi tam

Category	Theme	Sub-theme	Definition	Example Utterance
				olmuşsa öyle devam edebiliyorsam çok değiştirme taraftarı olmuyorum."
		Awareness of own-knowleadgeability	One has self-knowledge on how much one knows about solving algorithm question.	Eng: "I asked myself what do I know, what do I not know?" Tr: "kendime sordum ne biliyorum, ne bilmiyorum?"
	Other-knowledge	Instructor's approach	One has ideas on what instructors' approach may be during the course.	Eng: "Since the teaching style of one teacher is different from that of another teacher, this method was very good for both my teacher X and my teacher Y, of course, my teacher Z." Tr: "bir hocanın anlatma tarzı ile diğer hocanın anlatma tarzı farklı olduğu için bu yöntem X hocam da Y hocam da gayet iyi oldular Z hocam da tabii ki."
		Filling the form approach	One has ideas on used approach regarding filling the form.	Eng: "As a strategy, I started to learn a little more about what strategy is, thanks to the forms you provided." Tr: "Strateji olarak ise stratejinin ne olduğunu, sizin verdiğiniz formlar sayesinde biraz daha fazla öğrenmeye başladım."
		Peers' comprehensibility	One has ideas on how peers feel.	Eng: "My friend who designs algorithms in the way I do and makes the algorithm can understand. But my friend, who designs an algorithm in a very different style of his own, cannot understand." Tr: "Benim yaptığım tarzda algoritma tasarlayan, algoritmayı yapan arkadaşım anlayabilir. ama kendine özgü çok farklı tarzda bir algoritma tasarlayan arkadaşım anlayamaz."
Planning	Planning in advance	Task solution approach	One automatically plans the solution approach before solving the algorithm problem: considering shortest route, transforming flowchart to code block, using code blocks	Eng: "Then I think that it is necessary to pass the process through your own mind and transfer it to the paper and transfer that paper back to the computer." Tr: "daha sonra işlemi kendi zihninden geçirip kağıda aktarıp, o kağıdı tekrar bilgisayara aktarmanın olması gerektiğini düşünüyorum."
		Task objectives	One determines the objectives necessary for solving the problem before attempting. These objectives might be either course-based or anticipated new knowledge-based.	Eng: "Does the homework (programming output) fulfill the task given first?" Tr: "ödev ilk önce verilen görevi yerine getiriyor mu?"
	Interim planning	Task solution approach	One adapts the solution approach during solving the problem. Considering shortest route and self-questioning	Eng: "Sometimes when I use decision structures while designing, I wonder if it gets tiring. As mentioned in the question, I thought there might be differences in these aspects to see if there could be a shorter way." Tr:"bazen tasarlarken de hani karar yapılarını kullandığım zaman acaba yoruyor mu diye düşünüyordum. zaten soruda da belirtildiği gibi daha kısa yolu olabilir mi diye bu yönlerden farklılıklar olabileceğini düşündüm."
		Task objectives	One adapts the necessary objectives during solving the problem.	Eng: "So I said that if the assignment is usually about the if structure, I will use the if structure" Tr: "yani ödev genellikle if yapısı ile alakalı ise if yapısını kullanacağımı söyledim"
Monitoring	Monitoring of strategy use	Feeling-based	Used strategies based on students' feelings	Eng: "It had an impact on organizing my thoughts. I tried to calm myself down by sitting down and not getting completely stressed, saying okay, I'm taking a break, without straining myself. because after a while, morale starts to depress why I can't do it. in such things. I say to myself, can't I do it? when I sit down and start resting somewhere. sitting in my head. My thoughts are sitting in my head. The things I want to do actually fall into place more when I take a break when I am in pieces." Tr: "düşüncelerimi düzenleme açısından etkisi oldu. kendimi kasmadan tamam ben mola veriyorum diyerek, bir kenara oturarak tamamen strese girmeden, kendimi sakinleştirmeye çalıştım. çünkü bir yerden sonra neden yapamıyorum diye moral bozukluğu başlıyor. bu tür

Category	Theme	Sub-theme	Definition	Example Utterance
				şeylerde. hani olmuyor mu ben yapamıyor muyum diyorum kendi kendime. oturup bir yerde dinlenmeye başladığım zaman. kafamda oturuyor. düşüncelerim kafamda oturuyor. yapmak istediklerim aslında param parça bir şekildeyken mola verdiğimde daha fazla yerlerine oturuyor."
		Person-based	Used strategies based on students' own characteristics	Eng: "We were also using a counter, while we were adjusting the counter in algorithms, for example, we were determining the first number, the second number, the third number. When we gave i a value we wanted it to continue in numbers, I struggled a lot doing it and when I found my mistake, I tried a lot until I found my mistake. I finally found it." Tr: "bir de sayaç kullanıyorduk, algoritmalarda sayaç belirlemede onu ayarlarken örneğin birinci sayı, ikinci sayı, üçüncü sayı diye sayaç belirliyorduk. i'ye bir değer verdiğimizde onun devam etmesini istiyorduk sayılarda, onu yaparken çok fazla uğraştım ve hatamı bulduğum zaman, hatamı bulana kadar çok fazla deneme yaptım. sonunda da buldum."
		Course-based	Used strategies based on the course that students attend	Eng: "So when I watch the videos before the lesson, I definitely do it in the program. Just to see how it works, maybe I'm typing something wrong?" Tr: "Yani ders öncesinde videoları seyrederken muhakkak programda yapıyorum. Nasıl çalıştığını görmek için bide tabii acaba bir şeyleri yanlış yazıyor olabilir miyim?"
		Example-based	Used strategies based on different examples not available in courses	Eng: "If we haven't done it in the past classes, I'm looking for it on the internet." Tr: "Eğer geçmiş derslerde de yapmadıysak internetten araştırıyorum."
	Monitoring of progress	Error detection by reviewing the code	One detects error by reviewing the written rows	Eng: "Don't write the characters properly. using neat shapes. then when trying at the exit part. Does it give correct answers at all values? Like we found the truth by mistake. I was usually skimming, but I was falling for the character writing." Tr: "karakterleri düzgün yazma. düzgün şekiller kullanma. sonrasında da çıkış kısmında denerken. bütün değerlerde, doğru cevapları veriyor mu. yanlışlıkla mı doğruyu bulduk gibi. genelde gözden geçiriyordum ama karakter yazımında üstüne düşüyordum."
		Error detection by seeing the unwanted result	One detects error by seeing the result has missing value	Eng: "For example, because he said to enter a number on the screen. I entered the 1st came again. I entered three. I got seven. I hit zero. the answer was directly zero, that is, it tried to multiply by zero. It wasn't supposed to hit him." Tr: "mesela ekrana bir sayı gir dedi diye. girdim 1. bir daha geldi. üç girdim. yedi girdim. sıfıra bastım. cevap direkt sıfır oldu yani sıfır ile çarpmaya kalktı. onu çarpmaması gerekiyordu."
		Error detection by seeing the unworked result	One detects error by seeing the failure	Eng: "I couldn't see my mistake especially in grading. I saw it after running it." Tr: "özellikle notlandırmada yanlışımı görememiştim. çalıştırdıktan sonra gördüm."
		Error correction by questioning	One corrects the solution by questioning on the reasons of failure	Eng: "While changing it, the other one disappeared and there is something, there was a question mark in my head." Tr: "değiştirirken diğeri yok oldu da bir şey vardır, kafamda bir soru işareti de oldu."
		Error correction by revisiting the recorded videos	One corrects the solution by revisiting the recorded videos	Eng: "There were also times when my instructors shared their lectures and I watched them and corrected my mistakes." Tr: "hocaların ders kayıtlarını paylaştığında ve onları izleyip hatalarımı düzelttiğim de oluyordu."
		Error correction by revisiting the notes taken during watching	One corrects the solution by revisiting the taken notes	Eng: "I was trying to review the examples we did in class. I was just trying to find my fault." Tr: "derste yaptığımız örnekleri gözden geçirmeye çalışıyordum. öyle hatamı bulmaya çalışıyordum."

Category	Theme	Sub-theme	Definition	Example Utterance
		videos and course hour		
	Comprehension monitoring	Noticing lack of comprehension	One is aware of lack of comprehension regarding the algorithm elements	Eng: "When I did it with the method we always do, I couldn't get results, I couldn't get an answer. missing algorithm. I realized there had to be a change there." Tr: "her zaman yaptığımız yöntem ile yapınca sonuç alamadım, cevap alamadım. eksik algoritma oldu. değişiklik olması gerektiğini anladım orada."
		Demonstrating comprehension by differentiating	One differentiates between logical operators and algorithm problems	Eng: "There was a pretty big difference between the second and third. In the third, there was an assignment called multiplication until you hit zero for the first time. It was the first time such an assignment until the end, in short." Tr: "ikinci ve üçüncü arasında bayağı büyük fark vardı. üçüncüde ilk defa sıfıra basana kadar çarpma işlemi diye bir ödev vardı. sonuna kadar ilk kez böyle bir ödev oldu yani kısaca."
Evaluating	Evaluating learning outcomes	Checking correctness of the solution	One checks the correctness of the solution	Eng: "I'm assigning the name of the variable, let it be a variable. I distinguished variable s and ş. Just because I have to write s. For example, I'm going to assign a name to a variable. I will write a variable, not a variable. I assign this to my distraction as the income variable. In this case, there is an error in the program. so I look directly at the variables first, to see if I wrote it correctly." Tr: "değişkenin ismi atıyorum değişken olsun. degisken s ile ş yi ayırt etmiştim. s yazmam gerekiyor diye. mesela değişken diye bir isim atayacağım. değişken değil de, degisken yazacağım. bunu ben dalgınlığıma gelir değişkeni değişken diye atarım. bu durumda programda hata olur. o yüzden direkt değişkenlere bakıyorum ilk olarak, doğru mu yazmışım diye."
		Reflects on effectiveness of the solution	One reflects on the effectiveness of the solution	Eng: "I decided this was shorter, the second I did. I sent two solution methods while submitting the assignment." Tr: "bunun daha kısa olduğuna karar verdim, ikinci yaptığımın. iki tane çözüm yöntemi gönderdim ödevi gönderirken."
		Checking clarity of the solution		Eng: "I always paid attention to make it understandable rather than abbreviated." Tr: "kısaltmaktan ziyade anlaşılır olmasına dikkat etmiştim hep."
	Evaluating learning progress	Reflecting on self-efficacy	One reflects on self-efficacy after completing	Eng: "I haven't done anything like that until now, but first I determine it in my head, then I apply it, and then I evaluate myself whether it's really what I'm missing." Tr: "öyle bir şey yapmadım şu zamana kadar ama önce kafamda belirleyip sonrasında onu uygularım ve sonrasında gerçekten ne eksiğim var, gerçekten olmuş mu diye değerlendiririm kendimi."
		Reflecting on task demand	One reflects on task demand after completing	Eng: "After that, I opened the forms and put the algorithm directly next to it. Before I did this algorithm, I thought about what was going through my mind and wrote answers to the forms, in the scoring places." Tr: "ondan sonra formları açıp algoritmayı direkt yanına koymuştum. bu algoritmayı yapmadan önce aklımdan neler geçiyordu diye düşünüp formlara cevaplar yazmıştım, puanlama yerlerine."