Mining Epistemic Actions of Programming Problem Solving with Chat-GPT

Rwitajit Majumdar
Kumamoto University, Japan
majumdar@kumamoto-u.edu.jp

Prajish Prasad
FLAME University, India
prajish.prasad@flame.edu.in

Aamod Sane
FLAME University, India
aamod.sane@flame.edu.in

ABSTRACT

In programming problem-solving, learners engage in creating solutions to specific given task problems by writing an executable code. Prior research has shown that self-regulated learning (SRL) strategies help improve novice performance in solving programming problems. With the advent of Large Language Model (LLM) tools like ChatGPT, novices can generate reasonably accurate code by providing the problem prompt. They, hence, may forego applying essential self-regulation strategies such as planning and reflection to solve the problem. This research investigates if the above is the case. We designed a programming problem-solving task in an available online environment, LAreflecT. A set of self-regulation prompts was provided while learners could use ChatGPT to build their solutions. Learners' interactions with the elements in the LAreflecT and their generated artifacts are logged. We analyzed 42 undergraduate students' data and highlighted problem-solving approaches of groups with correct and incorrect end-point solutions through process mining and artifact analysis. The findings indicate when learners use LLM as support their epistemic actions involve refining their problem understanding and solution evaluation when supported with metacognitive prompts within the system. We discuss the reflections of the learners who had more than two conversations with ChatGPT and draw implications of designing SRL support while learning with generative AI.

Keywords

Programing Education, ChatGPT, LAreflecT, Process mining

1. INTRODUCTION

As students and instructors learned about Large Language Model (LLM) tools, many possible uses of such tools and worries about their use in Computer Science (CS) education surfaced in the community. In an early study, Lau and Guo [1] collected a snapshot of tool capabilities and instructor responses and proposed research opportunities for Computer Science Education Research (CSER), such as improved scaffolding, building educator-tailored tools, scalable assessments, feedback, and personalized instruction. Another early study [2] identified the possibilities of generating exercises, solution variants, and refactoring problems. These ideas for future work are being addressed in papers on exercise generation, code explanations, improvements of error messages [3, 4, 5], and so on. Such investigations of the possible uses of LLMs have emerged organically, driven by community perceptions of possibilities and worries.

R. Majumdar, P. Prasad, and A. Sane. Mining epistemic actions of programming problem solving with chat-gpt. In B. Paaßen and C. D. Epp, editors, *Proceedings of the 17th International Conference on Educational Data Mining*, pages 628–633, Atlanta, Georgia, USA, July 2024. International Educational Data Mining Society.

Loksa and Ko's proposed a framework of programming problem-solving and aspects of self-regulation [5]. Loksa and Ko proposed six distinct activities during problem solving - (1) Reinterpreting the problem prompt (2) Searching for analogous problems (3) Searching for solutions (4) Evaluating a potential solution (5) Implementing a solution, and (6) Evaluating an implemented solution. They also proposed five types of self-regulation that support problem-solving: (1) Planning, (2) Process monitoring, (3) Comprehension monitoring, (4) Reflection on cognition and (5) Self-explanation. The effects of LLM technologies use and how it fundamentally impacts programming problem-solving are still less investigated. In their extant form, they are likely to influence students' ability to regulate their programming problem-solving process. Why? – students will directly use LLM tools to generate code without sufficiently thinking about the problem. Need to be careful to understand the problem and evaluate the solution generated by LLM tools. Therefore, educators should ensure that educational interventions use these technologies in ways that foster self-regulated learning. Prasad and Sane [6], took Loksa and Ko's framework to develop SRL strategies while learners engage in programming activities with generative AI systems as support. Since LLMs automatically generate code, they provide the affordance to clearly separate the specification of the problem from the actual implementation during problem-solving. Hence, there is more emphasis on steps of coding problem understanding and solution evaluation. They provided examples of internalized thinking the students might have during the above two phases. Further, the types of self-regulation include not only managing concepts but also planning, monitoring, comprehending, and reflecting on the artifacts created for interaction (prompts) or generated by the LLM (responses).

While there exists a theoretical position of including LLM for specific programming problem-solving activities in the learning context, there was no empirical evidence of learners' interactions in such a learning context. We initiated this research from the perspective of learning analytics to investigate the learner behaviors given an online task environment that instantiated the above CS-ed theoretical framework. This work is guided by the following research questions:

What problem-solving strategies do students follow while attempting the programming task using an LLM tool like ChatGPT?

2. RESEARCH FOUNDATION

2.1 Using LLM tools in CS education

Since LLM tools work very effectively with languages, they make possible new kinds of educational interventions [7]. Sarsa et al. generate exercises using LLMs [8], finding that generated exercises are sensible and novel, and the problem statements are properly contextualized. They show that LLMs also generate suitable test cases. LLMs can generate code explanations [9] that are easy to understand [10] and helpful [11]. Other uses include improvements of error messages [12] that are sometimes better than the originals.

Perhaps the most interesting new aspect of LLM-based tools is that they allow conversation, enabling new variants of think-aloud problem-solving [10], and an unusual mechanized form of pair programming [11]. The idea of 'prompt engineering', the redesign of conversations is a LLM specific skill, is explored in the design of constraining tools like Promptly [12]. That overreliance on LLM-based tools is likely to affect SRL is discussed in [16].

2.2 Programing Problem-solving Framework

Loksa and Ko's proposed problem-solving activities into 3 broad phases - Problem Understanding, Use of Generative AI Models, and Solution Evaluation. In addition to Loksa and Ko's five types of self-regulation, Prasad and Sane's model also includes regulation strategies related to motivation and affect (bottom portion of Figure 1). The theoretical framework of Prasad and Sane [6] is taken as a foundation to create the learning task and analyze the interaction data collected during this study.

According to Prasad and Sane, Problem understanding is a key phase in the problem-solving process. A clear understanding of the problem is essential to build an adequate model of the task. An incorrect understanding can hinder all subsequent steps in arriving at a solution [5]. Several theories of metacognition and self-regulation highlight the importance of analysing the task, as well as understanding the surrounding context of the problem to be solved [15]. According to Flavell, the problem task can be perceived differently by a variety of learners, based on several parameters such familiarity/unfamiliarity, organization (well or poorly organized), redundancy and ambiguity in the task. Zimmerman highlighted the importance of setting appropriate goals and strategic planning as part of the task analysis phase in the self-regulation process [15]. In addition, Pintrich stressed the importance of prior content knowledge and metacognitive knowledge activation during the initial phases of problem solving. What are learners expected during the problem understanding phase? They plan and reflect on the problem context and the specification, and iteratively design prompts for the problem. These form the sub-categories of the problem understanding phase. It constitutes the sub-categories in the left part of Figure 1.

Evaluation and reflection on the final as well as intermediate solutions is an important aspect of problem solving. Zimmerman notes the role of self-evaluation in calibrating one's effort on a task with a standard or goal [16]. In the context of programming, this refers to evaluating whether the program meets the expected specifications of the Problem. The solution generated by generative AI models need not be ideal or appropriate in the first attempt. This makes evaluating the generated solutions an important activity in the problem-solving process. Learners are required to monitor and reflect on the solution and strategize subsequent problem-solving actions. The constituent elements of the solution evaluation phases involve effectively performing program comprehension and debugging, and also deciding when solutions generated by LLMs are hindering their progress.

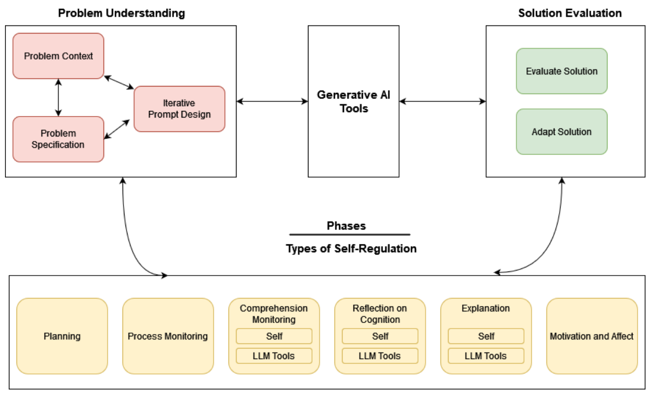


Figure 1. Self-regulation framework for programming problem solving using generative AI [6]

Table 1 highlights the suggested learning design and salient features of possible intervention. In this study, we aim to trace the learners' problem-solving process with reference to the above framework. Hence, we have instantiated the activity in an online learning environment that enables us to extract the trace data and analyze and interpret the process. The considerations in this specific study are reported in the last column of table 1.

Table 1. Educational Intervention Features according to [6] and how it is instantiated in this study.

Problem-solving skill	**Salient features of possible intervention [6]**	**Instantiation in this study**
Problem Specification	Provide specifications in modalities other than text	Provide a picture of one input and output scenario. Tracked the number of interactions the learner had, indicating they referred back to the problem specification.
Problem Context	Provide on-demand knowledge about programming concepts, constructs, and course topics	Provide this as an input element where learners reflect on what concepts they think are necessary to understand and solve the given problem specification
Problem Decomposition	Provide problems with multiple functions, classes, or concepts or with multiple sub-goals where learner develop the ability to combine and adapt multiple pieces.	A simpler problem was selected due to the time constraint of completing the task within the class hour. It was still at an appropriate level of complexity where different learners approached differently with variation in the correctness of the answers.
Prompt engineering	Provides a knowledge base of essential prompt categories.	This was an input element where learners shared their initial prompt and the link to their ChatGPT conversation. The conversation is analyzed as an artifact.
Program Comprehension and Evaluation	Provides an explanation for code	The learner could use ChatGPT to comprehend the generated code.
Self-regulation	Externalization of Metacognition and SRL strategies	The instructor could create an input element that prompted learners to externalize their metacognition, and they also write reflectively about their interaction with ChatGPT.

3. RESEARCH METHODS

3.1 Learning Context and Learning System

This particular pilot study is conducted in an undergraduate computer programming course offered to the students of a liberal arts college in a private university in India. The class was scheduled for 1 hour 55 minutes. The class included students in the age group of 19 to 20 years who were enrolled in their second year of study at the university. At the time of the research, this course was their second programming course after being introduced to Python in their first year.

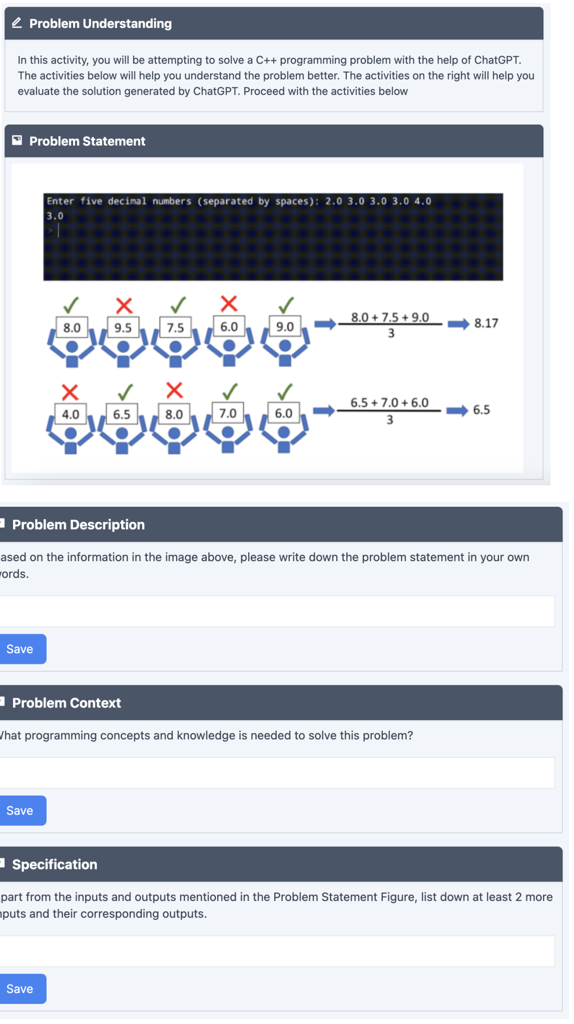
Learning Analytics' enhanced Reflective Task (LAreflecT) platform was used to conduct this study. It has two main components: an authoring tool where the teacher can create any microlearning activity and a viewer where the learner can attempt it. The platform can be linked to any learning management system (LMS) with a standardized learning tool interoperability (LTI) protocol. The user with a teacher role in the LMS has access to the authoring tool in the platform. Once they create an activity, they have to publish it to make it available in the activity viewer. The viewer has an activity attempt interface. It also logs the data of the users' interactions and then can visualize the data in the dashboard and also provides access for the teacher to download the dataset for further processing. This tool had appropriate affordances required for both instantiating the learning activity and providing access to the logged data required for this study.

The programming assignment was authored in the LAreflecT platform by the instructor. Learners were given a test case that contained an input and the corresponding required output. Learners were required to write code for that test case in LAreflecT. They could use ChatGPT for the activity, and the instructor created metacognitive prompts in the LAreflecT platform to reflect on their problem-solving process as defined in Table 1. Figure 2 presents the Problem statement and examples of input elements.

3.2 Data Collection and Analysis Method

An overall phenomenographic research approach [14] guided the research questions to focus on a single activity undertaken by the students enrolled in the course, the in-class programming assignment. The team of researchers, including the course instructor, then interpreted the different approaches that emerged from the interaction logs during that episode of the programming problem-solving.

We followed a purposive sampling and selected the students enrolled in the course (n=77, 39% females) as participants. The data extracted for this study included the LAreflecT logs during the programming class period in the two sections on September 15 and September 18, 2023. Of the total 77 students enrolled in this course, 16 students were either absent or dropped out of the course. 61 accessed the task on LAreflecT. 3 students didn't consent to the data sharing, so for this analysis, we considered 58 students' 2442 action logs as xAPI statements with three types of verbs. First *clicked* when any element is accessed and the accordion is opened, second, *answered* when a particular input element is answered by the user, and the third *time spent* when an opened according to is closed. The input by the students, as recorded in the logs, provided the artifacts of their initial problem definition, their first prompt to ChatGPT, their final solution, and their perceptions of ChatGPT. The instructor evaluated the final code for its correctness from 42 students whose log had it. In that dataset, there were 22 cases of learners whose final answer was correct (CA group) and 20 cases of learners whose final answer was incorrect (IA group).


Problem Understanding

In this activity, you will be attempting to solve a C++ programming problem with the help of ChatGPT. The activities below will help you understand the problem better. The activities on the right will help you evaluate the solution generated by ChatGPT. Proceed with the activities below

Problem Statement

```
Enter five decimal numbers (separated by spaces): 2.0 3.0 3.0 3.0 4.0
3.0
>
```

Problem Description

Based on the information in the image above, please write down the problem statement in your own words.

Save

Problem Context

What programming concepts and knowledge is needed to solve this problem?

Save

Specification

Apart from the inputs and outputs mentioned in the Problem Statement Figure, list down at least 2 more inputs and their corresponding outputs.

Save

Figure 2. The LAreflecT viewer interface for the learning task. The problem statement and example prompt for problem understanding

To answer RQ, the prominent interaction process for each group while they attempted the activity was extracted using the fuzzy miner algorithm implemented in DISCO (Fluxicon, 2021). The process map was created from event logs of answering the different elements in the overall problem-solving episode for the two different groups. While conducting process mining by using a fuzzy miner algorithm implemented in DISCO, the criteria for selecting an activity node (operations) of the process and the paths (linking consecutive operations) need to be set. The analysis parameters were selected following available references in the literature earlier [16]. It aims to balance the details of the emerging process path, which can be interpretable in the context and not abstract enough just to present independent loops of operations. To understand the overall attempt patterns, we filtered for only *answered* verbs which included the learners' actions on the input elements of the LAreflecT activity. All the 9 input elements were considered. The top 25% of prominent paths were visualized on the map.

4. FINDINGS

4.1 Task attempts of groups with correct and incorrect answers

Figure 3 presents the process map of both CA group and IA group regarding the frequency of cases and Figure 4 presents the time performance of the answering process. The salient differences are highlighted by number and discussed subsequently.

While the task was arranged in more of a linear flow in the LAreflecT interface, the process mining highlights the sequence of answering which involves updating different elements while problem-solving. In the Problem Understanding phase of the activity, problem description was updated by 50% of CA-group learners who got their final solution correct in contrast to 36% of IA-group learners (see highlight 1 in fig 3.). 31.8% of CA-group learners updated the problem description after reflecting on the concepts required to solve the problem in contrast to 20% in the IA-group (highlight 2 in fig 3). There were 22.7% of CA-group learners who updated the problem description after writing the prompt for ChatGPT. This was not a significant process path in the IA-group.

In the solution evaluation phase, 45.5% of the CA-group learners updated the prompts after evaluating the solution (highlight 3 in Fig 4.), and 22.7% updated the evaluation also (highlight 5 in Fig 3.). However, such pathways were not significant for the IA-group.

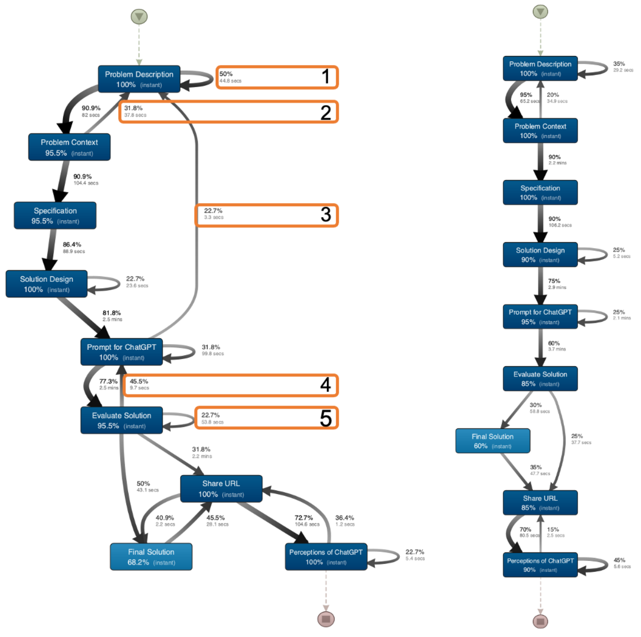


Figure 3. Interaction pattern of the groups in terms of case frequency. (left) Group with correct answer (CA group, n=22) (right) Group with incorrect answer (IA group, n=20)

Considering the time spent during the process, it gives an indicator of effort spent to input or reflect on before committing the answer to a specific element. It is seen that 31.8% of the CA-group learners spent an average 2.2 minutes before sharing the link of their ChatGPT conversation (see highlight 6 in Fig.4) in comparison to 37.7 sec for the IA-group learners. On the other hand, the IA-group learners' interactions around the Prompt for ChatGPT were longer than the CA-group (see highlight 7 in Fig 4.)

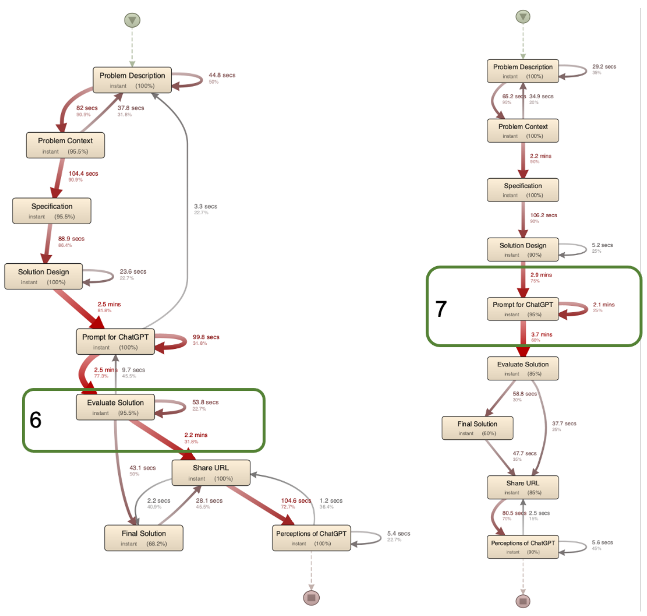


Figure 4. Interaction pattern of the groups in terms of time performance. (left) Group with correct answer (n=22) (right) Group with incorrect answer (n=20)

4.2 Interpretation along with qualitative data

Based on the end-point task accuracy of the learners, we explored the answering patterns in LAreflecT tool. The answering action given the prompts designed by the instructor, represented the learners' metacognition while problem-solving. For instance, reflecting on what are the concepts required while answering the *Problem Context* element required them to activate prior knowledge during the problem understanding phase. We found that the group that successfully got to the solution had significant pathways of refining their problem understanding (highlight 1,2,3). Before having multiple conversations with the ChatGPT tool, they spent considerable time formulating the initial prompt. This confirms that engagement in the problem-understanding phase is still necessary and is related to the final success of finding the programming solution. For solution evaluation, the analysis indicates that a proportion of members of the CA-group (highlight 4,5,6) followed that pathway where revisions were done by spending more average time than the IA-group. These patterns highlight that even if the LLM tool provides some response as a solution, it requires the learners to verify more whether the answer provided is valid, without which nearly 48% of the students didn't find the correct solution even while aided by ChatGPT.

Members of the IA-group who used LLM tools to have conversations longer than two interactions also spent more time before they submitted the evaluation of the solution. We looked at the response to the *evaluate solution* element in LAreflecT and found that utterances indicate that the students were aware the solution is incorrect because their prompts were not clear or incorrect. ["*No, I didn't (evaluate the solution). Because I was ineffective in presenting it. Make it clearer.*" - studentid_4931; "*I am not sure if the solution is correct by that is mainly because I am not sure my question/ problem statement is correct.*" studentid_5121]. While in two other cases, the learners were not sure of the solution provided by the LLM as it seems complex to them due to the use of programming elements that they still didn't know of ["*I think it is correct, but it is more detailed than I assumed it would be. It solved the problem differently than I the way I imagined it, making use of array concepts*." studentid_5314;

"*The solution seems correct. It looks like ChatGPT has understood what needs to be done and has used the logic that will give us the output. Since the code uses while loops, I cannot fully examine whether it is correct or not simply because I myself don't know how to execute them in C++. Maybe I can edit the prompt, asking it to use functions and conditionals."* studentid_4981]. In the last case, the learner was aware that the solution was not correct ["*Not entirely correct. yes there are incorrect parts. like it is not doing average only with the picked 3 numbers but it is taking the combinations of 3 numbers and doing average for it. so it is divided with 10 at the end."* studentid_3782*].* Only studentid_4981 among them had the problem description correct, while all others also formulated the problem incorrectly to begin with. On the contrary, all the CA-group people had the problem description correct, and while evaluating the solution, they reflected that ChatGPT's solution was often initially not the desired one or too complex ["*The first response or the first code it gave me is wrong. The solution code is correct, but it used too many complex functions and methods. I will ask Chatgpt to improve the code and use the most basic concepts of C++" studentid_4869].*

Members of the CA-group who used ChatGPT multiple times also reflected on the *Perception of ChatGPT* element in LAreflecT for a relatively longer time and had the following responses. One of them was critical about the LLM tool itself ["*It got the code correct but it will be using styles much different that individuals use themselves. It attempts to be more precise than the input tells it to thus making the code bulky. It usually gets the code correct but messes up simple calculations itself such as outputting 18/3 as 6.1."* studentid_4874]. Two believed ChatGPT was useful but made learning task simple [*"useful, but makes everything too convenient."* studentid_4814; "*Chatgpt made my work very easy. The code it gave me is correct as I checked it on replit. It made wrong when I just gave it my requirement as it made many complex assumptions but after I specified my actual method, it gave me the right solution. ChatGPT can be very helpful in programming tasks as it makes our task very simple"* studentid_4869]. One of them was aware of the solution themselves and wanted to use ChatGPT for more complex problem ["*If the question would have been more difficult and then I would nicely be able to judge chatgpt, here as I knew the answer, there was not a problem which I was asking chatgpt to solve"* studentid_5300*].* The last learner in the group had a detailed reflection on their own confidence of the programming problem-solving task as well as the role that was delegated to the LLM tool. Further, a reflection of how to use the tool for future learning was also written. ["*ChatGPT helped me implement logic that I had already figured out but in a programming language that I'm not too familiar with. I knew the concepts I would need for this problem -- arrays and sorting. However, I had little experience implementing those concepts in C++. ChatGPT thus wrote the C++ code for arrays and sorting for me. It got the logic right based on the prompts I gave. I'm not sure if it got the syntax right--I'm trusting that it did, since I'm not too familiar with C++ arrays and sorting. It also added some functionality that I didn't ask for -- it wrote code that prints out the sorted array before printing out the final answer. I can use ChatGPT for help with programming tasks that I'm stuck on. In the academic realm, ChatGPT can help me figure out the logic for a problem, write the code, and debug code i may already have written. Outside academics, ChatGPT helps me implement a lot of Python code for data analytics work on a variety of projects. I often give it elaborate problem statements and ask for implementations in Pandas and Numpy. I know enough of pandas and numpy to completely understand its solutions too. If i'm invested in learning a concept because I need it for a test, I wouldn't directly ask ChatGPT for the solution. Instead, i ask it to teach me/talk me through the logic. I prefer to write the code myself if I'm learning a concept for a test. Meanwhile, i often just need to get things done on data analytics freelance projects, so I spend more time tweaking the finished code that ChatGPT gives me. In both cases, i make sure i understand the code."* studentid_3313*].*

5. CONCLUSION AND FUTURE WORK

This study is one of the first exploratory ones that investigated programming problem-solving interactions to mine and interpret epistemic action sequences while working with ChatGPT. We adopted a prior framework from CS education [6] which proposed interventions to include LLM tools like ChatGPT that might influence learners' self-regulation skills. The empirical evidence through exploratory data analysis highlights those learners whose endpoint solution was correct emphasized refinement during the problem-understanding phase, while the group whose endpoint solution was incorrect often fixated on the use of the tool to iterate and find the solution but did not succeed in that specific episode. However, the responses from the IG group highlight that even when the end-point solution was incorrect, as a learning activity including interactions with ChatGPT helped them to reflect on their process and evaluate the solution provided by the LLM tool. The findings encourage using the self-regulated learning framework in computing education [6] by which activity and the designed prompts provide enough reflection opportunities for learners' self-regulation even when using LLM tools like ChatGPT. With authoring platforms like LAreflecT there is the scope to include such self-regulation prompts or navigation constraints to elicit learner reflection while learning programming problem-solving. We plan to explore the effect of using the dashboard in LAreflecT that can show the learners' attempt pattern and their self-reflections on the problem-solving task.

There are various limitations at this stage of the exploratory study. The given problem statement might not be complex enough as expected in the framework, but still, there was variation in the learner's end-point performance as well as their process behaviors that were interesting to report. Also, the overall process of attempting the answer is captured through log data, which does not capture the prior conceptual understanding of the students, their motivation, or any of their prior dispositions [17]. These might affect how the student engaged in the task and their behaviors. Further studies are required to investigate differences in student's programming problem-solving behavior in groups formed based on other relevant learner attributes, such as their self-regulated learning attributes. Finally, the participants in the context were liberal arts majors who participated in the programming course. The answering behaviors and interaction process might vary for participants from different specialization groups, academic levels, or activity contexts (MOOC courses, adult education scenarios, hackathons, etc), which requires further investigation.

6. ACKNOWLEDGMENTS

This work was partially supported by JSPS KAKENHI Grant-in-Aid for Scientific Research (B) JP22H03902.

7. REFERENCES

[1] Lau, S. and Guo, P., 2023, August. From" Ban it till we understand it" to" Resistance is futile": How university programming instructors plan to adapt as more students use AI code generation and explanation tools such as ChatGPT and GitHub Copilot. In *Proceedings of the 2023 ACM Conference on International Computing Education Research-Volume 1* (pp. 106-121).

[2] Finnie-Ansley, J., Denny, P., Becker, B.A., Luxton-Reilly, A. and Prather, J., 2022, February. The robots are coming: Exploring the implications of openai codex on introductory programming. In *Proceedings of the 24th Australasian Computing Education Conference* (pp. 10-19).

[3] Leinonen, J., Denny, P., MacNeil, S., Sarsa, S., Bernstein, S., Kim, J., Tran, A. and Hellas, A., 2023, June. Comparing code explanations created by students and large language models. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (pp. 124-130).

[4] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. 2022. Automatic generation of programming exercises and code explanations using large language models. In Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1. 27-43

[5] Dastyni Loksa and Amy J Ko. 2016. The role of self-regulation in programming problem-solving process and success. In Proceedings of the 2016 ACM conference on international computing education research. 83-91.

[6] Prasad, P. and Sane, A., 2024, March. A Self-Regulated Learning Framework using Generative AI and its Application in CS Educational Intervention Design. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (pp. 1070-1076).

[7] Stephen MacNeil, Andrew Tran, Dan Mogil, Seth Bernstein, Erin Ross, and Ziheng Huang. 2022. Generating diverse code explanations using the gpt-3 large language model. In Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 2. 37-39

[8] MacNeil, S., Tran, A., Hellas, A., Kim, J., Sarsa, S., Denny, P., Bernstein, S. and Leinonen, J., 2023, March. Experiences from using code explanations generated by large language models in a web software development e-book. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (pp. 931-937).

[9] Leinonen, J., Hellas, A., Sarsa, S., Reeves, B., Denny, P., Prather, J. and Becker, B.A., 2023, March. Using large language models to enhance programming error messages. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (pp. 563-569).

[10] Kelly YL Ku and Irene T Ho. 2010. Metacognitive strategies that enhance critical thinking. Metacognition and learning 5 (2010), 251-267

[11] Brian Hanks, Sue Fitzgerald, Ren√©e McCauley, Laurie Murphy, and Carol Zander. 2011. Pair programming in education: A literature review. Computer Science Education 21, 2 (2011), 135-173

[12] Paul Denny, Juho Leinonen, James Prather, Andrew Luxton-Reilly, Thezyrie Amarouche, Brett A Becker, and Brent N Reeves. 2023. Promptly: Using Prompt Problems to Teach Learners How to Effectively Utilize AI Code Generators. arXiv preprint arXiv:2307.16364 (2023)

[13] Denny, P., Prather, J., Becker, B.A., Finnie-Ansley, J., Hellas, A., Leinonen, J., Luxton-Reilly, A., Reeves, B.N., Santos, E.A. and Sarsa, S., 2024. Computing education in the era of generative AI. *Communications of the ACM*, *67*(2), pp.56-67.

[14] Jan Larsson, and Inger Holmström. "Phenomenographic or phenomenological analysis: Does it matter? Examples from a study on anaesthesiologists' work." International Journal of Qualitative Studies on Health and wellbeing 2.1 (2007): 55-64.

[15] Zimmerman, B.J., 2000. Attaining self-regulation: A social cognitive perspective. In *Handbook of self-regulation* (pp. 13-39), Monique Boekaerts, Paul R. Pintrich, Moshe Zeidner, Ed., Academic press, Cambridge, USA.

[16] Majumdar, R., Bakilapadavu, G., Li, J., Chen, M.R.A., Flanagan, B. And Ogata, H., 2021, November. Analytics of Open-Book Exams with Interaction Traces in a Humanities Course. In *ICCE 2021: 29th International Conference on Computers in Education Conference Proceedings Volume I* (pp. 352-361). Asia-Pacific Society for Computers in Education (APSCE).

[17] Stupnisky, R.H., Renaud, R.D., Daniels, L.M., Haynes, T.L. and Perry, R.P., 2008. The interrelation of first-year college students' critical thinking disposition, perceived academic control, and academic achievement. *Research in Higher Education*, *49*, pp.513-530.