*JKAU: Comp. IT. Sci.*, **Vol**. **8** No. 2, pp: 13 – 31 (1440 A.H. / 2019 A.D.)
Doi: 10.4197/Comp. 8-2.2

# The Impact of Guided Metacognitive Feedback on Novice Programmers Using Learning by Teaching Environment

**Ahoud Alhazmi, Rafika Maaroufi** and **Abdulwahab Aljubairy**

*Umm Al-Qura University, Al-leith, Makkah, Saudi Arabia*

*Abstract.* Learning-by-teaching is a powerful approach that enhances students to think deeply, orally and repeatedly. Several computer-based systems have been implemented where students play the teacher role and virtual agents play the tutee role. The existing systems focus on various domains, but none of them has considered programming problem solving. Additionally, the majority of these systems did not provide metacognitive support. They only focus on providing feedback as correct answers, and this type of feedback is called knowledge of correct response. However, this paper explores the influence of guided metacognitive feedback on novice programmers in a teachable agent environment. For that, a computer-based learning environment is built to enable the novice programmers to teach programming problem solving to an animated agent. It combines learning-by-teaching technique and metacognitive support in order to assist those beginners to acquire comprehensive learning on how to solve unfamiliar problems and prepare those programmers for future learning tasks. We conduct an experiment to compare the effect of the aforementioned feedbacks on the novice programmers' performance in learning-by-teaching paradigm. The results show that the metacognitive feedback has positive effect on novice programmers' achievement of solving problems. In addition, providing metacognitive feedback as explicit feedback in learning-by-teaching paradigm improves the novices' abilities to estimate what they know and what they do not know about how to solve new programming problems.

*Keywords*: Intelligent learning system, Learning-by-teaching approach, Metacognitive feedback Teachable agents, Teachable agent learning environment.

## 1. Introduction

Education is currently undergoing a revolution due to the advancement of educational technologies. It is not enough to provide hardware and software support to students in order to assist the acquisition of a domain knowledge. More effort needs to be concentrated on how to foster students' skills and be independent learners [1]. In the time being, educational establishments are starting to focus on differentiated learning to shift the education approach from a one-to-many homogeneous experience to a one-to-one deeply immersive and personalized learning experience [2].

For that, many researchers from psychology, education and computer science fields have worked together for creating educational software, where students can follow a curriculum individually and enhance their learning capabilities [3]. This type of educational software includes virtual agents that play pedagogical roles such as teachers, students or mentors. In traditional Intelligent Tutoring System (ITS), virtual pedagogical agents play the tutor role to teach a domain

knowledge to students and provide materials and exercises. These agents evaluate the level of students' learning. However, this type of ITS cannot promote students' motivation and skills. In contrast, Teachable Agent Environment (TAE) which is another type of educational software overcomes the limitation of traditional ITS. It comes from an educational technique called learning-by-teaching, where human students play the teacher role to teach an animated agent new information about a particular subject using well-structured visual representations [4][5]. Consequently, this technique assists the students to understand the materials effectively because they prepare themselves to deliver the information to the animated agent.

A number of TAEs have been built for many domains. These systems affect positively on student's learning processes at any education level [6]. These systems also motivate students to spend more time and effort to achieve their goals from teaching the agents [7] [8] [9]. None of these systems has considered programming problem solving domain. They only focus on providing support to the human students on their cognitive skills in a particular domain.

Although students need cognitive skills to acquire knowledge, they also need metacognitive skills to develop their abilities to regulate independently their learning process. Metacognition refers to thinking about one's own thinking process such as study skills, memory capabilities, and the ability to monitor the learning process [10]. Furthermore, it plays a significant role of the distinction between experts and beginners in programming domain [11].

The preparation stage for teaching others will force students to organize their ideas to gain a deeper understanding of the materials and know about their strengths and weaknesses in order to be able to explain the materials in a simple way for others [12]. Learning by teaching encourages to improve the metacognitive skills implicitly. However, Halpern [13] recommends that the metacognitive support should be explicit to induces the students to improve their metacognitive skills. This paper discusses the effect of the guided metacognitive feedback as explicit feedback within learning-by-teaching system. For that, we build a computer-based learning environment that combines learning-by-teaching approach with metacognitive support to assist novice programmers to develop and enhance their monitoring abilities and programming skills.

This system provides a solution to novice programmers who are frustrated when they do not know how to deal with unfamiliar programming problems because they usually tend to focus on writing the code rather than understanding the problem properly. Especially, most introductory programming courses often focus on the features of the programming language including syntax [14]. Students in these programming courses acquired cognitive skills because they are usually taught about basic cognitive skills such as how to create variables, selection statements and iterative statements. These courses do not help students how to think, evaluate and monitor their skills for solving problems in an efficient approach. Mastering only cognitive skills is not enough for beginner programmers to deal with unfamiliar problems because they need to know when to use it [15]. This aspect of the problem is called metacognition. Using metacognitive skills helps novice programmers about when and how to use specific strategies for problem solving [16].

The remaining sections are organized as follows: Section 2 explains metacognition and the essential need of it on the process of solving programming problems successfully. Section 3 presents how our work is different

from the existing learning-by-teaching systems. Section 4 is dedicated to explaining our system. Then, Section 5 discusses the experiment setting. In Section 6, we present the findings of the experiment, and Section 7 discusses these findings. Section 8 presents the limitations of our study.

## 2. Metacognition and Programming Problem Solving

Metacognition is a higher order thinking that allows students to understand, analyse and control their own thought processes [17] [18]. It is as an awareness about students to learn or solve a problem because it allows them to plan and monitor their learning process to improve their performance [19]. For that, it is an important factor that distinguishes between poor and good learners. Borkowski et al. in [20] argue that good learners are more flexible in their methods to deal with problems and they are more self-monitoring. As well, those students have a large repertoire of strategies and they are more organized.

The contents of metacognition come from the learner's internal mental representations (such as skills about cognition) but the contents of cognition come from the learner's external reality [17]. In other words, metacognitive strategies guarantee the achievement of the goals whereas cognitive strategies help learners to achieve the goals [21]. For example, when we read a text, we need cognitive skills. This is different when we want to monitor our own understanding of the text and that is considered metacognitive skills. Another example, novice programmers know basic cognitive skills such as how to create constants and variables; comparison and logical operators; selection statements; iterative statements; and arrays. When they have to know when and how to use them, that is considered metacognitive skills.

### *A. Metacognitive and Cognitive Feedback*

Feedback is one of the most powerful influences on learning and achievement. It is conceptualized as information provided by an agent (e.g., teacher, peer, or self) regarding aspects of one's performance or understanding. The aim of providing feedback is to assist students to develop and increase their knowledge, skills or understanding.

Several studies have examined which a good type of feedback has to be provided to students. For instance, when the feedback response tells if the answer is correct or incorrect, it is less useful to learners. However, providing the feedback with the correct answer is useful than previous type of feedback. The latter type of feedback is called Knowledge of Correct Response (KCR) [22]. Brown and Knight in [23] discovered that the feedback should be with less comments and give more guidance to enable students to think more about how to fix the mistakes and use their intellect skills. The definition of characteristic of intelligent novices is the ability to control and monitor their own thought processes.

Intellectual skills are defined as methods that can be used by an individual to evaluate and organize information [24]. Jacobse and Harskamp in [25] demonstrate that students who utilize their intellectual skills have well-developed metacognitive abilities. Hence, improving the performance of novice programmers necessitates supporting and guiding their metacognitive skills.

Metacognitive feedback is a guided feedback that assists programmers to be aware of their own problem-solving skills and their understanding of the activities they have to do. Thus, such feedback enables novice programmers to experience their strengths and weaknesses.

### *B. The Difficulties of Novice Programmers and the Essential Metacognitive Skills*

Novice programmers' approach for solving a programming problem is reading the problem statement and starting directly writing the code. That means they are confused between problem solving and coding. Additionally, they usually use the trial-and-error strategy.

By using this strategy, novice programmers may provide a correct solution, but it might be not the ideal solution. In addition, their code would be unreadable and hard to understand. For example, if novice programmers read this problem "Write a pseudo-code to check each number between 1 to 99 and determine the odd numbers and print them". They may understand only printing odd numbers hence they output all odd numbers by adding 2. That would print the correct results, but this solution shows that the novices does not consider all the required information in the problem statement which is to check if the number is odd or not. Another example is "Write an algorithm to calculate $X^y$ ". Novice programmers could provide a partially correct answer if they consider y is a positive number. That means they did not examine all special cases when y equals zero or a negative number. Indeed, the limitations of their approach is using line-by-line of written code rather than understanding the big picture of the program structures. Vickers in [26] emphasizes that the problem solving is the heart of programming rather than coding. For that, in this research, we focus on how to solve programming problems before starting coding.

As mentioned previously, those programmers usually learn basic cognitive skills in their classes. However, they face a problem about how and when to use cognitive skills. Mayer in [15] discuss that novice programmers need to master metacognitive skills to solve unfamiliar problems. Metacognition skills distinguish between novice and expert programmers. Eteläpelto in [11] reported that expert programmers (i.e. who had at least two years' experiences of the field) were superior to the novice programmers (i.e. who were trained in the domain but did not have any work experience) only in the metacognitive knowledge of the programming task. Expert programmers have more abstract hierarchical organization based on the principles of program functions while novice programmers have only syntax-based knowledge organization.

Furthermore, novice programmers lack the mentoring goal-setting process, planning the solution, designing and awareness of errors in the solution [27]. Thus, providing metacognitive support in these phases will assist novice programmers to become strategic learners and expert programmers. One part of our system will be responsible for providing metacognitive feedback to a human student in order to develop their metacognitive skills.

### C. Thinking about the Process of Solving Programming Problems

The first attempt at thinking about the process to solve a problem was provided by George Polya in [28]. He displayed the main important steps that assist solving any problem successfully. These steps are: understanding the problem, devising a plan, carrying out the plan and looking back. The process of these steps is not linear because the problem solvers need to go back and forth to the steps until they reach the goal.

Vickers in [26] suggests some strategies for each stage in Polya's approach because he claims these strategies would assist the programmer to get rid of confusion and concentrate on problem solving rather than writing the code. Figure 1 shows what most

strategies that would be useful for programming problems in our system

## 3. Previous Works of Teachable Agent Systems

Several TAEs have been built for many domains such as river ecosystems, math, economics, biology and history [2] [5] [8] [29] [30]. None of TAEs has considered programming problem solving domain. Furthermore, learning-by-teaching has three phases: preparing to teach, teaching process and gaining feedback. The first two phases are existed in all of the previous TAEs whereas the third phase is not included in some TAEs. Gaining feedback is essential since it helps human students who take the tutor role to evaluate their own understanding of the domain knowledge and evaluate their approach that is used to transfer this understanding to the teachable agents. We believe including these three phases in an educational environment make a system more effective.

As mentioned previously, providing only cognitive feedback would not assist students to prepare themselves for future learning. Therefore, two TAEs were implemented to provide metacognitive support to human students such as selecting the appropriate problems [8] [29] but this support did not help those students to improve their own skills of problem solving. Throughout our research, we take care of the development of problem-solving skills.

## 4. The Teachable Agent Learning Environment

Two virtual pedagogical agents (Amy and Ms. Sarah) are developed to interact with a novice programmer in the interest of providing the interactive learning environment. Amy takes the tutee role where she can learn from the tutor (the novice programmer) about programming problem-solving strategies. Based on the knowledge that Amy obtained from the tutor during the teaching stage for a particular problem, she can apply this knowledge to other isomorphic problems in the quiz stage and try to solve them by herself. Therefore, that can show the quality of the knowledge that the tutor passes to Amy. Also, she can provide explanations about what she understood, and she can ask her tutor when she could not understand some parts of the solutions. The second agent is Ms. Sarah who plays the mentor role. She grades Amy's solutions and provides guided metacognitive feedback to Amy's tutor. She can also evaluate Amy's performance for solving the problem.

Figure 2 shows the summary of the learning cycle in the system. It has five components as follow: presenting the selected problem, teaching Amy, observing Amy's solution of another isomorphic problem, obtaining metacognitive feedback from Ms. Sarah and correcting Amy's solution to get the ideal solution.

### *A. The Architectural Design of the System*

We translated all the requirements of the system into the architectural design as shown in Figure 3. As mentioned earlier, the system has two animated agents to interact with a novice programmer to provide interactive learning environment. Similar to this type of system, other agents are needed to handle data [8]. Tracker agent and Interfacer agent are designed to handle data relevant to the specified functionalities. They are intermediate agents between the human students and two animated agents when some events are generated. These four agents are described in the following subsections.

#### *1. Amy's Brain*

To make an efficient learning-by-teaching environment, the agent Amy needs to demonstrate the qualities of an active student. She can ask questions if she does not

understand. She also provides an explanation of her understanding after the human student teaches her again about her mistakes or missing tasks to obtain a profound understanding from the interactions with a tutee.

Furthermore, Amy behaves as a beginner programmer who usually knows the basic cognitive skills in the programming structures such as variables and constants; logical and comparison operators; selection and iterative statements. She also knows the declarative knowledge of each problem. For example, she knows the base of the hexadecimal is 16 and binary is 2. Another example, she knows the equation for checking the number is odd or even. The human teachers usually do not provide the declarative knowledge in the quiz for solving any programming problem because they expect that the students know it before. However, she learns from the tutor about the procedural knowledge of the problem. The definition of procedural knowledge is the action sequence for solving problem [31].

### *2. Ms. Sarah's Brain*

Ms. Sarah is the mentor agent which is an expert in the problem-solving domain. She provides guided metacognitive feedback to support the human students to improve their metacognitive skills. In addition, she evaluates Amy's solution and she also asks the Amy's tutor to help Amy for providing a correct solution. As known, any programming problem has more than one solution. For that, Ms. Sarah has the ability to consider all the situations for every solution.

### *3. Tracker*

It uses an automated approach to determine the human student's progress and then notifies the other animated agents (Amy and Ms. Sarah). For example, if the human student requested Amy to take a quiz without teaching her, the tracker detects that and trigger an event to inform Amy. Another example, if the human student spent a long time to teach Amy, in this case, another type of trigger will be issued to Ms. Sarah to provide some help.

### *4. Interfacer*

The Interfacer agent acts as a broker between all the system structures and the reasoning mechanism and the Graphical User Interface (GUI). In the beginning, the human student will interact with the GUI. Then, Interfacer will display the updated view in the GUI. When Amy and Ms. Sarah need to interact with the human student by exchanging messages (i.e. feedback or explanation), the Interfacer will display that message to the human student.

## *B. Overview of the Activities of the Learning Environment*

We design the system's modules (as listed below) to facilitate human students' learning and to stimulate their learning interest. Some of these activities are done by the human students. Other activities are provided by our system and the animated agents.

- **Module 1.** Selection of the problems – human students can use it to select a specific problem that is provided from the system to be taught to the teachable agent (Amy).
- **Module 2.** Teaching workspace - it has all required keywords that can be used in writing a pseudo-code of solving any problem (Figure 4).
- **Module 3.** Amy's feedback and her queries - it is to present Amy's feedback or her queries to her tutor. Also, the tutor can answer to her queries.
- **Module 4.** Quiz workspace- it is to enable Amy to take a quiz and she tries to solve another problem
- **Module 5.** Ms. Sarah's Feedback - it is to grade and evaluate what Amy does in the

quiz. Ms. Sarah also provides guided metacognitive feedbacks (Figure 5).

- **Module 6.** Re-teaching - based on the Ms. Sarah's feedback, the tutor can re-teach Amy to get the ideal answer.

- **Module 7.** Explanation - it enables Amy to provide an explanation of what she understands from the tutor after fixing her mistakes.

- **Module 8.** Indicator - it enables to show the quality of the solution of the current problem.

- **Module 9.** Skillmeter - it is to display Amy's overall progress.

- **Module 10.** Previous answers - it presents the tutor and Amy solutions for the attempted problems.

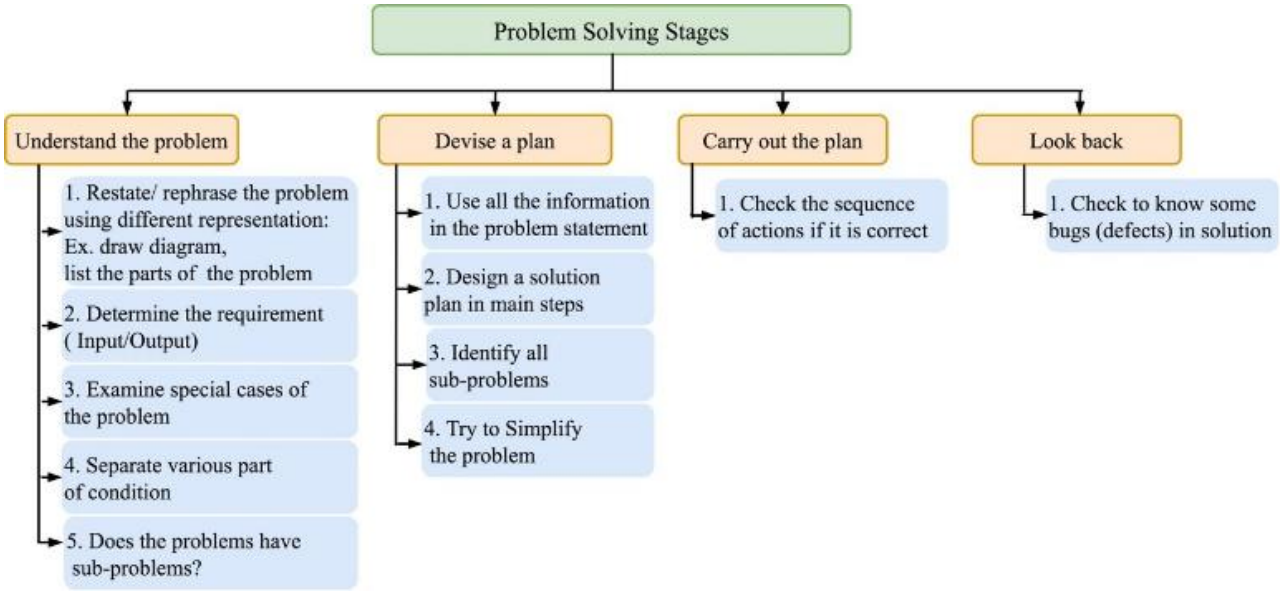


**Fig. 1. Problem solving stages based on Polya's approach and some strategies for each stage based on [26].**

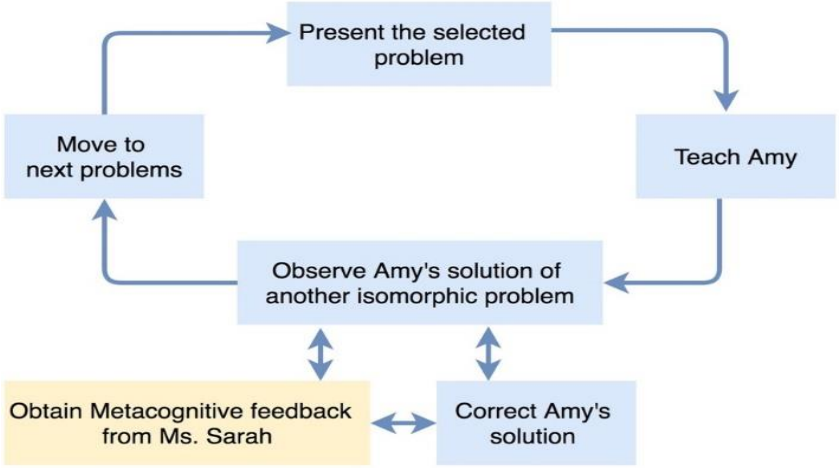


**Fig. 2. Learning cycle.**

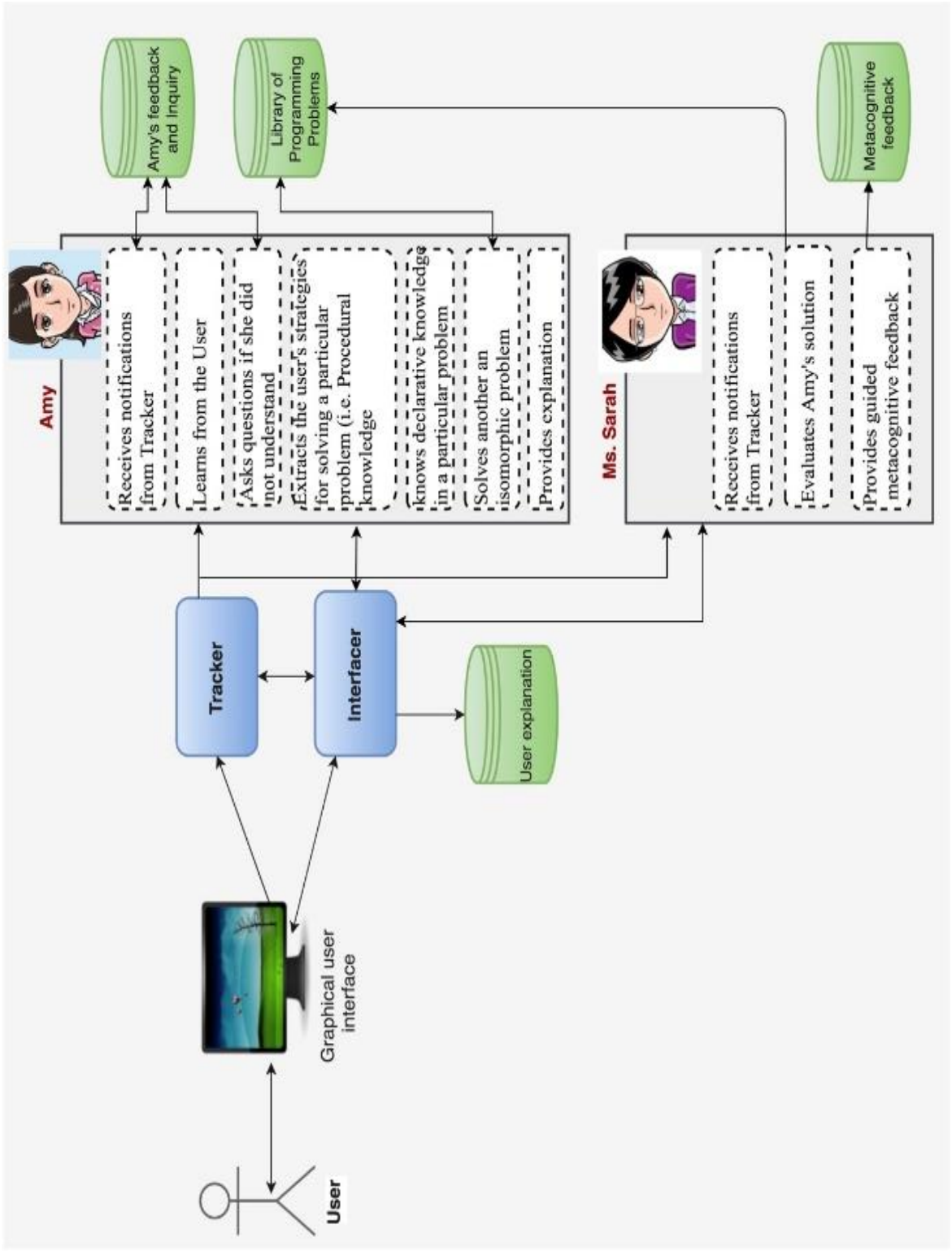


**Fig. 3. The architectural design of the System.**

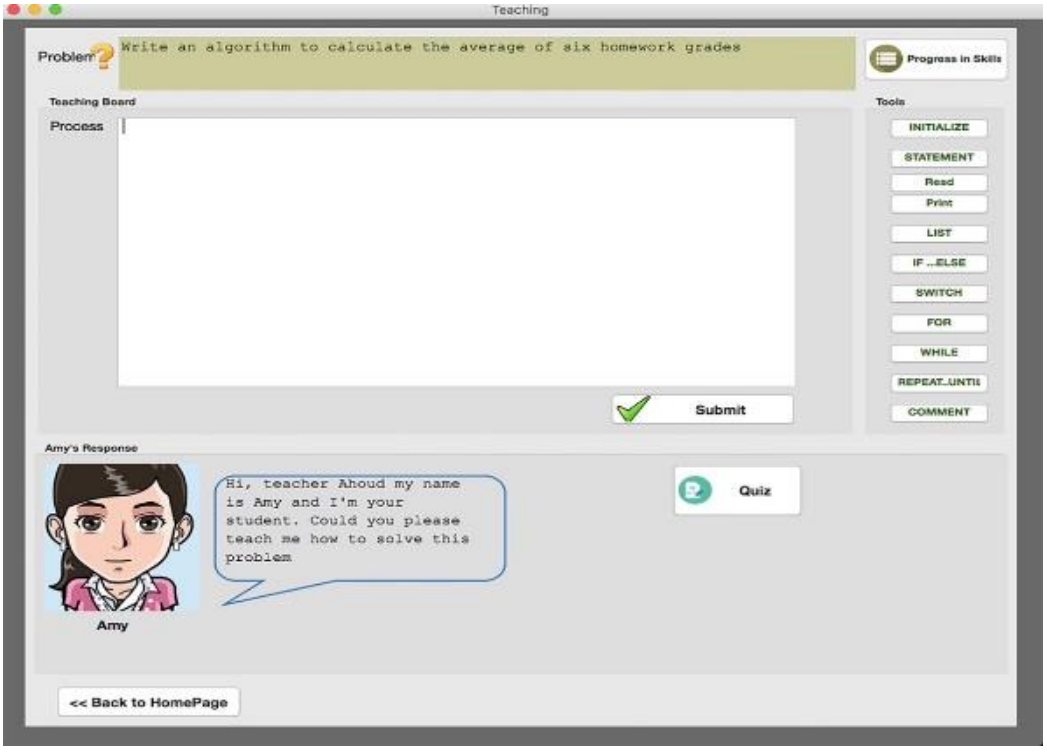


**Fig. 4. Graphical User interface in our system (the Teaching stage).**

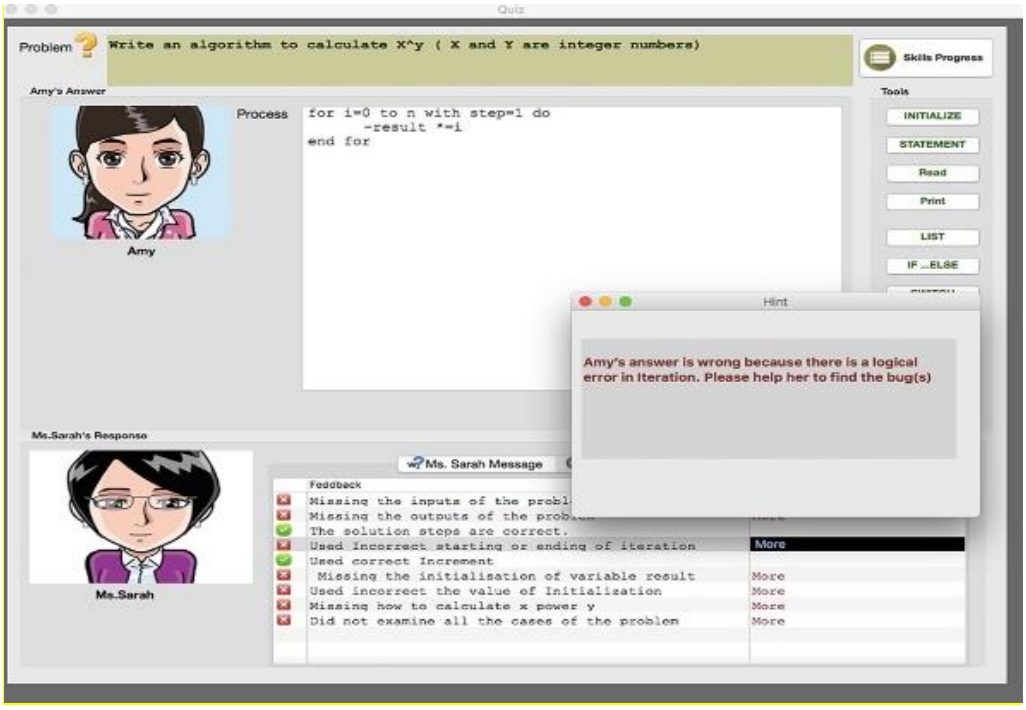


**Fig. 5. Some feedback from Ms. Sarah in the Quiz stage that shows the issue in Amy's Solution.**

### *C. The Context of the Chosen Isomorphic Problems*

We develop 16 pairs of programming problems. One pair of problems will be taught to Amy by her tutor and another will be answered by Amy in the quiz stage. Our development of the problems is based on the near-isomorphic problems such as structured problem representation, the sequences of process and number of the cases. However, in some cases the difference between two problems is not a big deal in our research because we want to focus on the human student's thinking about the problem-solving strategies as known as metacognitive skills rather than the cognitive skills. Table 1 shows some pairs of programming problems in both stages.

**Table 1. Some examples of isomorphic programming problems.**

| | *Problems in Teaching Stage* | *Problems in Quiz Stage* |
|---|---|---|
| 1 | Write an algorithm to calculate N! | Write an algorithm to calculate $X^y$ (X and Y are integer numbers) |
| 2 | Write a pseudo-code to convert binary to the decimal number and print the number (not including floating-point number) | Write a pseudo-code to convert hexadecimal to the decimal number and print the number (not including floating-point number) |
| 3 | Write an algorithm to find the largest of an unsorted set of integers | Write an algorithm to find the smallest of an unsorted set of integers |

### *D. Guided Metacognitive Feedback*

Ms. Sarah has the responsibility to provide guided metacognitive feedback as mentioned above. This feedback is organized as successive hints that provide the answer to the current problem step based on the strategies that discussed previously in Section 2.3. Ms. Sarah's feedback contains two stages. Firstly, she provides clues about the reasons for the incorrect answer of Amy. Secondly, she suggests specific activities to the human student to assist Amy. Table 2 shows some examples of Amy's behaviour and what Ms. Sarah's feedback will be provided based on this behaviour.

**Table 2. Several examples from Ms. Sarah's Feedback based on Amy's behavior.**

| | *Amy's behaviour* | *Ms. Sarah Feedback to human tutor* |
|---|---|---|
| 1 | Did not know the subject such as: how to convert binary to decimal | Amy's answer is wrong because she does not know how to convert binary number to decimal number. You should read about it and then re-teach her this information. |
| 2 | Forgot some special case such as 0! | Amy's answer is incomplete because this problem has more than one case. Please read the problem statement and think about all the other cases and then help her. |
| 3 | Did not separate various parts of conditions. Such as xy if y >0 or y <0 has another solution | Amy's answer is incomplete because this problem has more than one condition. Please help her to find the other conditions. |
| 4 | Missed some information in the problem statement such as forget test if a number is odd or not | Amy's answer is incomplete because she does not use all the information in the problem statement. Please read the problem statement and find missing information and then help her. |
| 5 | Did not break down the problem as (step by step) such as in matching string no using two pointers | Amy does not know how to break down the problem correctly there is something missing. Please help her to break down the problem correctly |
| 6 | Missed one of main process such as initialization | Amy's answer is not complete. There is one of the main steps is missing. Please review her process and help her on this step. |
| 7 | Incorrect ordering of action such as Result = L*W<br>Read L*W | Amy's answer is wrong because the ordering of action is not correct. You should help Amy to order the actions correctly |

## 5. Materials and Methods

This research compares between metacognitive feedback and KCR feedback in learning-by-teaching paradigm. Our research questions are:

1. What are the influences of KCR and metacognitive feedback on novice programmers' achievement and the metacognitive skill of the knowledge monitoring?

2. How can KCR and metacognitive feedback affect on novice programmers to adjust their approach for teaching the teachable agent (Amy)?

### *A. Experimental Design and Participants*

The experiment design has two groups: experimental group and control group. The experimental group use the system which provides guided metacognitive feedback from Ms. Sarah in the quiz stage. The Control group use similar system but, Ms. Sarah in this system provides only the ideal answer.

The number of participants is 21, and they are second-year undergraduate students of Computer Science. They are from College of Computer at Al-lieth in Umm Al-Qura University. All of them are female and their ages range between 18 and 25 years. They studied and passed the Introduction to Computer Science course in the first year. This course offers the foundations of computer science as well how to design simple algorithms. Currently, they are studying a computer programming course. In this course, students are subjective to explore the features of a programming language. The experiment is conducted on the third and fourth week when they started studying this course. We exclude the students in the third year or above because they may have experienced the programming problems provided in our system. We also exclude the first-year students because they have not studied yet any programming course. Those participants are divided randomly into two groups: 11 participants in the experimental group and 10 participants in the control group.

### *B. Procedural Skills Tests*

The experimental design has three procedural skills tests: pre-test, post-test, and delayed-test. The aim of these tests is to assess participants' proficiency in programming problems. Each test has two parts. The first part is to estimate students' knowledge in terms of whether or not they would be able to solve a given problem by answering "Yes" or "No" to the question. For example: "Do you think you can solve the given problem correctly?" After completing the first part, the participants will do the second part which requires to solve programming problems using pseudo-code.

Regarding the description of containing the three tests, they provide isomorphic programming problems. We develop these problems into three constructs for structured program: sequence, selections and repetitions. The pre-test and post-test include four problems: (one problem of sequence (i.e. simple instructions), two problems of selections (two-way selection and multiple selections) and one problem of repetition). Also, the delayed-test includes four problems: (one problem of sequence (i.e. simple instructions), one problem of selections (two-way selection) and two problems of repetition).

The problems in delayed-test was challenging because we increase the number of repetition problems. We believe the repetition of constructs needs more control and monitoring skills. Especially, they also are a combination of sequences and selections of constructs. Note, the problems in the three tests are different from the problems in our TAE.

### *C. Structure of Study*

The study is conducted in three separate sessions to measure any long-term retention of acquired skills. There are many tasks should be completed within two hours for the first two sessions. Each task should be completed in a particular duration. On session one, the participants fill up a personal and educational background and they also complete the pre-test. In addition, participants watch a video that explains how to use the system. Then, they are free to use our system in order to teach Amy some programming problems. The second session is after four days from the first session. The participants continue to teach Amy other programming problems. After that, participants are given 40 minutes to answers post-test. The Delayed-test is administered one week after the second session for 40 minutes.

### *D. Interaction Logs*

In order to achieve our aim of examining the impact of our system on the novice programmers' skills, we implement log files. The system automatically records the interactions of the human student events in these log files. Every action from opening the frame until the closing is recorded. Three log files are maintained to recode the human student interactions for selecting the problems in homepage, teaching stage and the quiz stage.

## 6. Results

### *A. The Effect of both feedbacks on Novice Programmers*

To answer the first research question, we verify the results by comparing the pre-post tests and pre-delayed tests of both groups. We mark each question for each participant similar to [21] [32] approach as follows: 2 points if a participant provided the ideal solution for the question, 1 point if the participant provided a partially correct solution, 0.5 point for incomplete solution and 0 for incorrect solution or if the participant did not provide any solution. Then, the total for all the questions in each test is calculated.

To compare the pre-test and post-test for participants in the experimental group, Figure 6 shows the performance for those participants in both tests. We find the performance of 82% of the participants in the experimental group has improved. However, we find the performance of only 30% of the participants in the control group has improved as shown in Figure 7.

To compare the pre-test and delayed-test for participants in the experimental group, Figure 8 and Figure 9 show the performance of both groups in pre-test and delayed-test. We find the performance of 70% of the participants in the experimental group has improved but the performance of only 50% of the participants in the control group has improved.

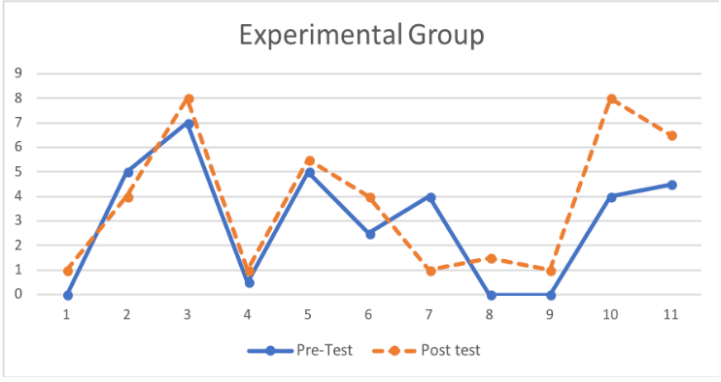


**Fig. 6. Comparing the Score of Participants for Both Tests (Pre-Test and Post-Test) in the Experimental Group.**

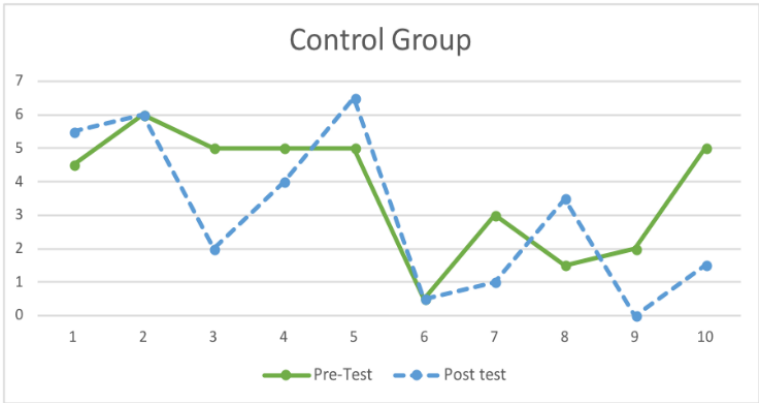


**Fig. 7. Comparing the Score of Participants for Both Tests (Pre-Test and Post-Test) in the Control Group.**

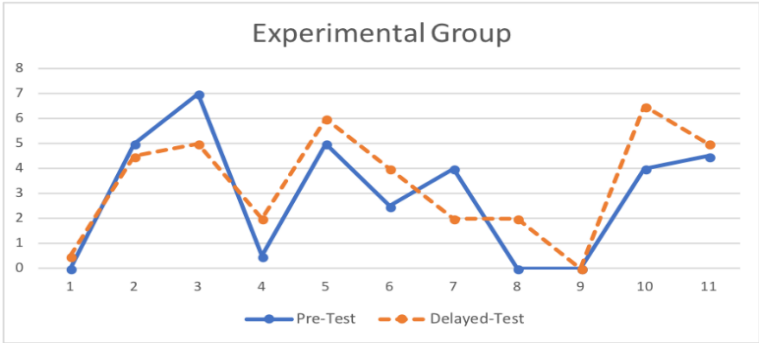


**Fig. 8. Comparing the Score of Participants for Both Tests (Pre-Test and Delayed-Test) in the Experimental Group.**

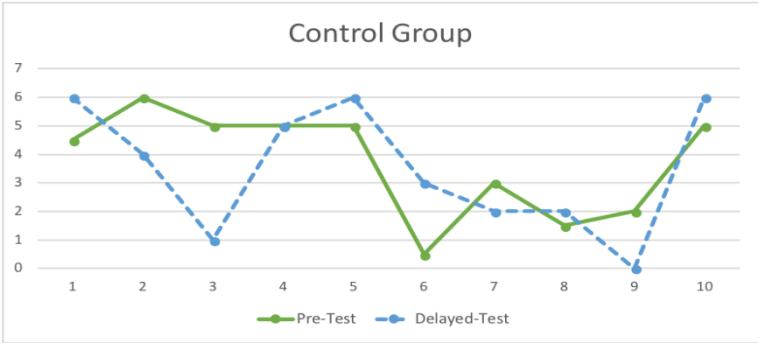


**Fig. 9. Comparing the Score of Participants for Both Tests (Pre-Test and Delayed-Test) in the Control Group.**

Table 3 shows the mean and standard deviation of student' tests. We choose T-test to examine how the performance of novice programmers varied before and after using learning-by-teaching system without considering the two conditions of feedback in the beginning. We find the t-value is -0.1 for the pre-post tests for all participants without considering the two conditions. The mean of post-test is greater a bit than the mean of the pre-test and we find the level of confidence for the one-tail test of significance is 55.1%. In

addition, we find the t-value is -0.4 for the pre-delayed tests. The mean of delayed-test is greater than the mean of the pre-test and the level of confidence for the one-tail test of significance is 66.8%.

With considering the two conditions of feedbacks, we find the t-value is -0.7 for the pre-post tests of the experimental group. The mean of post-test of the experimental group is significantly greater than the mean of the pre-test of the same group and we find the level of confidence for the one-tail test of significance is 75.9%. Furthermore, we find the t-value is -0.8 for the pre-delayed tests of the experimental group. The mean of the delayed test of the experimental group is significantly greater than the mean of pre-test and we find the level of confidence for the one-tail test of significance is 78.4%.

In contrast, for the control group, we find the t-value is 0.7 for the pre-post tests. That means, the mean of post-test of the control group is lesser than the mean of the pre-test and we find the level of confidence for the one-tail test of significance is 76.3%. In other words, the performance of the control group after the post-test is not improved significantly. Furthermore, we find the t-value is 0.3 for the pre-delayed tests. The mean of pre-test is still significantly greater than the mean of delayed-test and we find the level of confidence for the one-tail test of significance is 60.6%.

**Table 3. Mean and standard deviations for the three procedural skills tests.**

| *Condition* | *Measurement* | *Pre-test* | *Post-test* | *Delayed-test* |
|---|---|---|---|---|
| **Experimental group** | Mean | 3.0 | 3.8 | 3.8 |
| | Standard deviation | 2.5 | 2.9 | 2.2 |
| **Control group** | Mean | 3.8 | 3.0 | 3.5 |
| | Standard deviation | 1.9 | 2.4 | 2.2 |
| **Overall** | Mean | 3.3 | 3.4 | 3.6 |
| | Standard deviation | 2.2 | 2.6 | 2.2 |

### *1. The Impact of Programmers' Prior Knowledge on Solving Problems*

We divide the participants into three clusters (HPK, LPK and APK) based on their pre-test scores (as shown in Table 4) because we want to investigate who have improved her performance in both groups. We find only 3 participants have improved in the control group in the post-test. Those participants belong to APK cluster whereas participants in both HPK and LPK did not improve. Conversely, in the experimental group, we find that participants, whose scores improved in the post-test, are from the three clusters. Interestingly, two of the participants from APK cluster achieved scores as the participants in the HPK cluster and one participant in the LPK cluster achieved scores as the participants in the APK cluster.

**Table 4. The numbers of participants based prior knowledge.**

| *Groups based prior knowledge* | *Numbers of participants* | | |
|---|---|---|---|
| | ***HPK*** | ***APK*** | ***LPK*** |
| ***Scores*** | ***6 to 8*** | ***4 to 5*** | ***0 to 3*** |
| **Experimental Group** | *1* | *5* | *5* |
| **Control Group** | *1* | *5* | *4* |

### *2. Changing in Knowledge Monitoring Accuracy (KMA)*

After getting the results of the participants' performance in solving the problems, we want to measure the changing of their metacognitive skill on knowledge monitoring. We have used a method for measuring the knowledge monitoring accuracy (KMA) based on [33]. Tobias and Everson's in [33] assessment instrument evaluates the student's knowledge monitoring ability by first asking him/her whether he/she is able to solve a problem and later asking him/her to solve that problem. The KMA results from the match between these two pieces of information. That would assist us to assess novice's knowledge and understanding.

We use the two parts in Procedural tests (as discussed in Section 5.2) to assess the changing in knowledge monitoring. We calculate the means of KMA scores in each test for each participant based on Tobias and Everson's formula for scoring the KMA. Then, we measure the means of the KMA scores for both groups among these tests. The average of KMA of the experimental group in pre-test is -0.5, in post-test is -0.182 and delayed-test is 0.138. While the average of KMA of the control group in pre-test, post-test and delayed-test is -0.1, -0.138 and 0.163, respectively. As it can be seen, the mean of the score of the KMA of the experimental group has improved in post-test while there is no improvement in the control group on their monitoring knowledge.

### *B. The Effect of both feedbacks on Teaching Amy*

For the second research question, we analyse the performance of teaching Amy by looking at the log files of each participant. We mark the solutions that are provided to Amy for the selected problems during the Teaching stage. Our approach of marking is as presented in section VI.A Then, we measure the average score for each problem in log files for both groups.

Most participants taught Amy eleven problems. The challenges of the problems increase from the first to the last problem in the system gradually. However, participants did not teach Amy remaining problems because of the time limitation. Figure 10 shows the average of the performance for each problem of both groups. There is a little bit increase on the performance of the experimental group than the control group. That means the experimental group has acquired some strategies such as looking back, monitoring and control for teaching Amy as well as solving the problem. The performance of the control group was close to the experimental group because the learning-by-teaching strategy can engage the metacognition in an implicit way.

## 7. Discussion

Learning-by-teaching assists learners to understand the domain knowledge in a profound way. Students who teach their peers perform better on a quiz than those who prepare only themselves for taking the same quiz [34]. Our findings show most novice programmers in our experiment improve their achievements for solving programming problems during the three tests regardless of the type of the feedback they have received. This result supports the efficiency of learning-by-teaching paradigm in improving student's skills in problem solving [29]. Furthermore, our results show all participants spent all the time to teach Amy and use the system. This agrees with previous research [7] [8] [9] which report this type of paradigm motivates students to spend more time and effort with animated agents.

As mentioned previously, having well-developed metacognitive thinking skills assist novice programmers to improve their performance on solving programming problems. We believe learning-by-teaching technique can encourage the practice of metacognitive skills in an implicit way, but it could not induce novice programmers to practice metacognitive skills in an effective way. This study investigates the impact of providing metacognitive feedback as an explicit support on novice programmers using learning-by-teaching environment. The results show the metacognitive feedback has a positive effect on the novice programmers' performance more than KCR feedback in learning-by-teaching paradigm, and this is similar to the previous research conducted by Halpern [13]. Thus, the metacognitive feedback can enhance the novice programmers to think

about one's thinking and earn the ability to know their strengths and weaknesses.

Furthermore, our findings show metacognitive feedback is more beneficial for all prior knowledge ability of beginner programmers. Specifically, low prior knowledge programmer benefits from metacognitive feedback than KCR feedback. This result agrees with the previous finding that shows metacognitive support are more beneficial for who have low prior knowledge than high prior knowledge [35].

We are not interested only in the impact of both feedback on novice programmers' achievement, but we also are interested in examining the changing of their metacognitive skill of knowledge monitoring. Our results show novice programmers who received metacognitive feedback improve their abilities to monitor their knowledge. They become more aware of their knowledge and their abilities of dealing with unfamiliar programming problems.

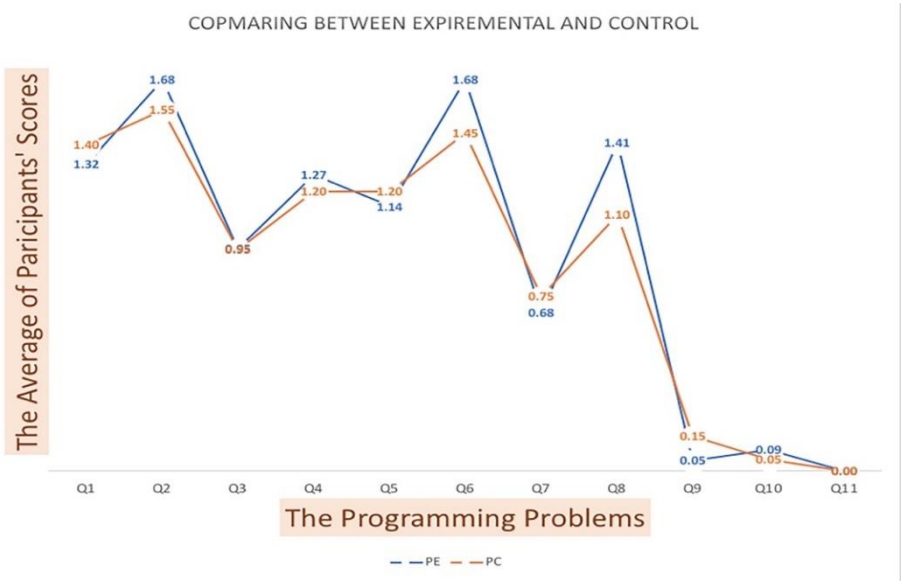


**Fig. 10. Comparing the average of eleven problems in log files for both groups. Hints PE: refer to the experimental group and PC to the control group.**

## 8. Limitations

We recognize our experiment has several limitations. One limitation is number and gender type of participants in the experiment. It was very difficult to gather enough volunteers for the experiment.

Allocating participants into the two groups randomly is another limitation. We should have tested the participants' knowledge of programming problem-solving in advance before the allocation. Fortunately, we did not face this issue in our experiment.

There is also a limitation which is not related to the experiment, but to the design of the system. It is the numbers and types of programming problems are presented in the system. They were not tailored to each student's skills of programming problem-solving. Thus, some participants found these problems in the system are more challenging during the experiment. However, the majority of participants tried to solve them even if they did not provide complete solutions. Thus, we believe that some of the results may have been influenced by these limitations.

The designing of educational systems is an interdisciplinary work encompassing computer science, education, and psychology fields. We as computer scientists needed to read more about the other two fields. It would be better if we worked closely with education and psychology experts in order to make sure that we meet all requirements for designing the system for novice programmers.

## 9. Conclusion and Future Work

In order to enhance the metacognitive skills of beginner programmers in the three difficult stages (formulating the problem, planning the solution, designing the solution), we build computer-based system that combines learning-by-teaching technique and guided metacognitive support. The aim of this research is to investigate the influence of providing metacognitive support as explicitly support on novice programmers using learning-by-teaching paradigm. For that, we conduct an experiment to evaluate the effect of this feedback comparing with KCR feedback on novice programmers' problem-solving skills and their approach to teach the teachable agent (Amy). The results show the metacognitive feedback has a positive effect on the novice programmers' performance. In terms of improvements related to adjusting the novice programmers' approach for teaching Amy, the performance of the group who received metacognitive support is slightly higher than the other group. The analysis reveals that the former group acquire some strategies such as looking back, monitoring and control their teaching process, and judgements about their abilities to solve new problems.

Even though the results of this study reveal many interesting findings, it will be important to investigate carefully the relationships between learning styles and metacognitive support in the future work. For example, development of Ms. Sarah's brain to understand the behaviour of the human students and their learning style could assist to provide more suitable feedback to the human students. In addition, the sample size should be increased in the future work as well as the sample has to be heterogenous.

### *Acknowledgements*

We would like to thank Dr. Amali Weerasinghe for her helpful discussions. We would like to thank the Saudi Arabian Cultural Mission in Australia for their support. We would also like to thank participants for their voluntary work. Our experiment would not be conducted the without them.

**References**

[1] **Hacker, D.J., Dunlosky, J.** and **Graesser, A.C.** eds. (1998) Metacognition in educational theory and practice. Routledge.

[2] **Okita, S.Y.** and **Schwartz, D.L.** (2013) Learning by teaching human pupils and teachable agents: The importance of recursive feedback. *Journal of the Learning Sciences*, **22**(3): 375-412.

[3] **de Raadt, M., Toleman, M.** and **Watson, R.** (2004) Training strategic problem solvers. *ACM SIGCSE Bulletin*, 36(2): 48-51.

[4] **Biswas, G., Tan, J.** and **Schwartz, D.L.** (2006) January. Feedback for metacognitive support in learning by teaching environments. In: *Proceedings of the Annual Meeting of the Cognitive Science Society*, (Vol. 28, No. 28).

[5] **Kirkegaard, C., Gulz, A.** and **Silvervarg, A.** (2014) June. Introducing a challenging teachable agent. In *International Conference on Learning and Collaboration Technologies* (pp. 53-62). Springer, Cham.

[6] **Schwartz, D.L., Chase, C., Chin, D.B., Oppezzo, M., Kwong, H., Okita, S., Biswas, G., Roscoe, R.D., Jeong, H.** and **Wagster, J.D.** (2009) Interactive metacognition: Monitoring and regulating a teachable agent. *Handbook of metacognition in education*, pp:340-358.

[7] **Chase, C.C., Chin, D.B., Oppezzo, M.A.** and **Schwartz, D.L.** (2009) Teachable agents and the protégé effect: Increasing the effort towards learning. *Journal of Science Education and Technology*, **18**(4): 334-352.

[8] **Leelawong, K., Viswanath, K., Davis, J.M., Biswas, G., Vye, N., Belynne, K.** and **Bransford, J.D.** (2003) Teachable Agents: Learning by Teaching Environments for Science Domains. In: *IAAI* (pp. 109-116).

[9] **Pareto, L., Haake, M., Lindström, P., Sjödén, B.** and **Gulz, A.** (2012) A teachable-agent-based game affording collaboration and competition: evaluating math comprehension and motivation. *Educational Technology Research and Development*, **60**(5): 723-751.

[10] **An, Y.J.** and **Cao, L.** (2014) Examining the effects of metacognitive scaffolding on students' design problem solving and metacognitive skills in an online environment. *Journal of Online Learning and Teaching*, **10**(4):552.

[11] **Eteläpelto, A.** (1993) Metacognition and the expertise of computer program comprehension. *Scandinavian Journal of Educational Research*, **37**(3): 243-254.

[12] **Pishghadam, R.** and **Khajavy, G.H.** (2013) Intelligence and metacognition as predictors of foreign language achievement: A structural equation modeling approach. *Learning and Individual Differences*, **24**: 176-181.

[13] **Halpern, D.F.** (2014) Critical thinking across the curriculum: A brief edition of thought & knowledge. Routledge.

[14] *ACM IEEE Joint Task Force on Computing Curricula, Computing Curricula 2001, Computer Science* (2001) IEEE Computer Society, Association for Computing Machinery. p. 240.

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

[16] **Davidson, J.E., Deuser, R.** and **Sternberg, R.J.** (1994) The role of metacognition in problem solving. *Metacognition: Knowing about knowing*, pp: 207-226.

[17] **Flavell, J.H.** (1979) Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *American Psychologist*, **34**(10): 906.

[18] **Hogan, M.J., Dwyer, C.P., Harney, O.M., Noone, C.** and **Conway, R.J.** (2015) Metacognitive skill development and applied systems science: A framework of metacognitive skills, self-regulatory functions and real-world applications. In: *Metacognition: Fundaments, applications, and trends* (pp. 75-106). Springer, Cham.

[19] **Schraw, G.** and **Dennison, R.S.** (1994) Assessing metacognitive awareness. *Contemporary Educational Psychology*, **19**(4): 460-475.

[20] **Borkowski, J.G., Carr, M.** and **Pressley, M.** (1987) "Spontaneous" strategy use: Perspectives from metacognitive theory. *Intelligence*, **11**(1): 61-75.

[21] **Gama, C.A.** (2005) Integrating metacognition instruction in interactive learning environments, *Doctoral Dissertation*, University of Sussex.

[22] **Narciss, S.** (2013) Designing and evaluating tutoring feedback strategies for digital learning. *Digital Education Review*, **23**:7-26.

[23] **Brown, S.** and **Knight, P.** (2012) *Assessing Learners in Higher Education*. Routledge.

[24] **Donald, J.G.** (1985) Intellectual Skills in Higher Education. *Canadian Journal of Higher Education*, **15**(1): 52-68.

[25] **Jacobse, A.E.** and **Harskamp, E.G.** (2012) Towards efficient measurement of metacognition in mathematical problem solving. *Metacognition and Learning*, **7**(2): 133-149.

[26] **Vickers, P.** (2008) *How to Think Like a Programmer: Problem Solving for the Bewildered*. Cengage Learning EMEA.

[27] **Deek, F.P.** and **McHugh, J.A.** (2003) Problem solving and cognitive foundations for program development: an integrated model. *Proceedings of the Sixth International Conference on Computer Based Learning in Science (CBLIS)*, pp: 266-271.

[28] **Polya, G.** (1957) *How to Solve It Garden City*. NY Doubleday.

[29] **Matsuda, N., Keiser, V., Raizada, R., Tu, A., Stylianides, G., Cohen, W.W.** and **Koedinger, K.R.** (2010) June. Learning by teaching SimStudent: Technical accomplishments and an initial use with students. In: *International Conference on Intelligent Tutoring Systems* (pp. 317-326). Springer, Berlin, Heidelberg.

[30] **Nichols, David Martin** (1994) Intelligent Student Systems: An Application of Viewpoints to Intelligent Learning Environments. *Ph. D.*, Computing Department, Lancaster University, Lancaster, UK.

[31] **Rittle-Johnson, B.** and **Alibali, M.W.** (1999) Conceptual and procedural knowledge of mathematics: Does one lead to the other? *Journal of Educational Psychology*, **91**(1): 175.

[32] **Nurulain, S.** and **Rum, M.** (2016) *A metacognitive support environment for novice programmer using semantic web* (Doctoral dissertation, University of Malaya).

[33] **Tobias, S.** and **Everson, H.T.** (2002) Knowing What You Know and What You Don't: Further Research on Metacognitive Knowledge Monitoring. Research Report No. 2002-3. *College Entrance Examination Board*.

[34] **Frith, C.D.** (2012). The role of metacognition in human social interactions. *Phil. Trans. R. Soc. B*, **367**(1599): 2213-2223.

[35] **Luckin, R.** and **Hammerton, L.** (2002) Getting to know me: Helping learners understand their own learning needs through metacognitive scaffolding. In *International Conference on Intelligent Tutoring Systems* (pp. 759-771). Springer, Berlin, Heidelberg.

# تأثير الملاحظات ما وراء المعرفية على المبرمجين المبتدئين في بيئة التعلم عن طريق التدريس بواسطة الأنظمة المعتمدة على الكمبيوتر

**عهود الحازمي**، و **رفيقة معروفي**، و **عبدالوهاب الجبيري**

*جامعة أم القرى، مكة المكرمة، المملكة العربية السعودية*

*المستخلص*. التعلم عن طريق التدريس نهج قوي يعزز الطلاب للتفكير بعمق، شفويًا وبشكل متكرر. تم تنفيذ العديد من الأنظمة المعتمدة على الكمبيوتر، حيث يلعب الطلاب دور المعلم، ويقوم الوكلاء الافتراضيون بدور الطالب. تركز الأنظمة الحالية على نطاقات مختلفة، ولكن أيًّا منها لم ينظر في حل المشاكل البرمجية. بالإضافة إلى ذلك، فإن غالبية هذه الأنظمة لم تقدم الدعم ما وراء المعرفية. فهم يركزون فقط على تقديم التعليقات كإجابات صحيحة، ويسمى هذا النوع من ردود الفعل: معرفة الاستجابة الصحيحة. ومع ذلك، تستكشف هذه الورقة تأثير التعليقات أو الملاحظات ما وراء المعرفية الموجهة على المبرمجين المبتدئين في بيئة التعلم عن طريق التدريس. لذلك، تم إنشاء بيئة تعلم تعتمد على الكمبيوتر لتمكين المبرمجين المبتدئين من تدريس حل مشكلات البرمجية إلى طالب افتراضي. فهو يجمع بين تقنية التعلم عن طريق التدريس والتفكير ما وراء المعرفي، من أجل مساعدة هؤلاء المبتدئين في الحصول على التعلم الشامل حول كيفية حل المشاكل البرمجية غير المألوفة، وإعداد هؤلاء المبرمجين لمهام التعلم المستقبلية. تم أجراء تجربة لمقارنة تأثير تقديم التعليقات كإجابات صحيحة والتعليقات ما وراء المعرفية على أداء المبرمجين المبتدئين في بيئة التعلم بالتدريس. وتظهر النتائج أن ردود الفعل التفكير ما وراء المعرفية لها تأثير إيجابي على إنجاز المبرمجين المبتدئين لحل المشاكل. بالإضافة إلى ذلك، فإن تقديم ملاحظات ما وراء المعرفية بأسلوب صريح في نموذج التعلم بالتدريس يحسن قدرات المبتدئين على تقدير ما يعرفونه وما لا يعرفونه عن كيفية حل مشاكل البرمجة الجديدة.

*الكلمات المفتاحية*: نظام التعلم الذكي، التعلم عن طريق التدريس، ملاحظات ما وراء المعرفية، بيئة التعلم بالوكلاء الافتراضيين.