



UWL REPOSITORY

repository.uwl.ac.uk

A comprehensive skills analysis of novice software developers working in the professional software development industry

Mian, Imdad Ahmad, Ijaz-ul-Haq, Anwar, Aamir, Alroobaea, Roobaea, Ullah, Syed Sajid, Almansour, Fahad and Umar, Fazlullah (2022) A comprehensive skills analysis of novice software developers working in the professional software development industry. Complexity, 2022. pp. 1-12. ISSN 1076-2787

<http://dx.doi.org/10.1155/2022/2631727>

This is the Published Version of the final output.

UWL repository link: <https://repository.uwl.ac.uk/id/eprint/9265/>

Alternative formats: If you require this document in an alternative format, please contact: open.research@uwl.ac.uk







Copyright: Creative Commons: Attribution 4.0

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy: If you believe that this document breaches copyright, please contact us at open.research@uwl.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Research Article

A Comprehensive Skills Analysis of Novice Software Developers Working in the Professional Software Development Industry

Imdad Ahmad Mian ¹, Ijaz-ul-Haq ², Aamir Anwar ³, Roobaea Alroobaea ⁴,
Syed Sajid Ullah ⁵, Fahad Almansour⁶, and Fazlullah Umar ⁷

¹Department of Science, SZABIST University Islamabad, Islamabad, Pakistan

²Faculty of Education, Psychology and Social Work, University of Lleida, Lleida, Spain

³School of Computing and Engineering, University of West London, London, UK

⁴Department of Computer Science, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia

⁵Department of Information and Communication Technology, University of Agder (UiA), Grimstad N-4898, Norway

⁶Department of Computer Science, College of Sciences and Arts in Rass, Qassim University, Buraydah 51452, Saudi Arabia

⁷Department of Information Technology, Khana-e-Noor University, Pol-e-Mahmood Khan, Shashdarak 1001, Kabul, Afghanistan

Correspondence should be addressed to Syed Sajid Ullah; sajidullah718@yahoo.com and Fazlullah Umar; fazlullahumir@gmail.com

Received 1 May 2022; Revised 10 June 2022; Accepted 25 June 2022; Published 15 July 2022

Academic Editor: Muhammad Ahmad

Copyright © 2022 Imdad Ahmad Mian et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Measuring and evaluating a learner's learning ability is always the focus of every person whose aim is to develop strategies and plans for their learners to improve the learning process. For example, classroom assessments, self-assessment using computer systems such as Intelligent Tutoring Systems (ITS), and other approaches are available. Assessment of metacognition is one of these techniques. Having the ability to evaluate and monitor one's learning is known as metacognition. An individual can then propose adjustments to their learning process based on this assessment. By monitoring, improving, and planning their activities, learners who can manage their cognitive skills are better able to manage their knowledge about a particular subject. It is common knowledge that students' metacognitive and self-assessment skills and abilities have been extensively studied, but no research has been carried out on the mistakes that novice developers make because they do not use their self-assessment abilities enough. This study aims to assess the metacognitive skills and abilities of novice software developers working in the industry and to describe the consequences of awareness of metacognition on their performance. In the proposed study, we experimented with novice software developers and collected data using Devskiller and a self-assessment log to analyze their use of self-regulation skills. The proposed study showed that when developers are asked to reflect upon their work, they become more informed about their habitual mistakes, and using a self-assessment log helps them highlight their repetitive mistakes and experiences which allows them to improve their performance on future tasks.

1. Introduction

The development of software is a logically complex task. The main four components of software development and its value ability are (i) the process applied, (ii) the tool used in the development, (iii) stakeholders, and (iv) timeline and budget [1]. Producing quality software is the key component to the success of any software product. The implementation

of the best development process, quality characteristics, and sophisticated technology have immense effects on the quality of the software [2].

Software quality can be affected by two main components: (a) software process quality and (b) software product quality. Product quality is dependent on process quality [2]. The process quality has received more attention from the research community [3]. Whereas the implementation phase

of the development process has received less attention, particularly from a behavioral perspective [4, 5]. The implementation phase consists of coding, testing, installing, and supporting the system [6]. As mature tools are available for coding, testing, installing, and supporting (troubleshooting) the system, along with these tools, the programmer's ability and skills have additional importance and effect on the software quality.

The programmer's ability and skills can be commonly viewed as the experience he/she has, different certifications, products developed (portfolio), and communication skills of the individual person [7]. These are the common methods that are generally applied to check the talent and capability of a programmer and have positive measures [4]. However, these factors do not show the individual thinking ability or self-judgment of the programmer during development. To achieve this goal, it is important to measure individual metacognitive skills such as self-exploration, self-assessment, understanding from example (example-based learning) [8], and most importantly learning from their mistakes. These measures can have diverse effects on individual performance and therefore require an in-depth analysis to highlight their importance and influence on young software programmers.

Since metacognitive skills are used subconsciously, many people are either unaware of this phenomenon or unable to manage them properly. Research shows that people who use their self-assessment skills are more progressive than those who do not use their self-assessment skills. Normally, people ignore these types of learning skills in many learning environments. A programming job is a very mentally challenging job, where a programmer's intellectual abilities, knowledge, and cognitive and metacognitive skills are used aggressively. Different types of studies have been carried out to assess the metacognition skills of different learners in different academic domains. However, minimal work has focused on the industry and no such work has been proposed for young professionals working in the software industry.

Our research objective is to analyze the self-assessment skills of novice developers, which affect their development work and to evaluate their performance on awareness of using self-assessment. The research will focus on

- (1) Analysis of metacognitive abilities and their usage by novice software developers.
- (2) Correlation of different types of novice developer groups and their performance on the basis of their improvement/diminishment.

To resolve these issues, the following research question is to be addressed:

- (i) What role can self-assessment play in the lives of novice software developers in minimizing repetitive mistakes?

To answer this main question, we will ask further (sub) research questions as follows:

- (i) What are self-assessment skills?

- (ii) How self-assessments skills help people perform better?
- (iii) How can self-assessment skills help novice software developers in the field of programming?
- (iv) What is the effect of self-assessment on novice developers' work?
- (v) What is the difference between developers who use self-assessment and those who do not?
- (vi) What kinds of mistakes a novice developer makes without analyzing their work?

2. Related Work

As like metadata can be defined as data about data, similarly metacognition can be defined as cognition about cognition [2]. Metacognitive learning involves specific knowledge of knowing that what are the different aspects of individual actions and what are the relations between these aspects that affect the learning of that individual [9]. Research on metacognition found that by applying different methods and awareness of metacognition, the learner's learning ability can improve significantly. Özsoy and Ataman [10] argue that metacognition practices are very important for successful learning strategies for teacher training, students' learning, and content development [11]. Özsoy and Ataman [10] conducted a study on students with mathematical disabilities and revealed the importance of metacognition in adulthood. He concluded that some students overestimated and some underestimated their mathematics results. The study on metacognition presents two different terms of mistake, which affect the learning process, made by the learner during learning: (1) Underuse: which means not asking for help when the subjects need help and (ii) Overuse: the learner asks for help repeatedly while not using his own mind to think about some problem [12].

2.1. The Social Cognitive Theory (SCT). In the paper [13], the authors relate the different factors of person metacognition as personal factors, behaviour, and environmental influence. The author [13] relates these factors and concludes that there is a relation between these components which affect the metacognition of students. These factors described in Figure 1 describe the social cognitive theory (SCT). The SCT describe that:

- (i) The interaction between the person and their behaviour is influenced by their thoughts and actions [14].
- (ii) The interaction between the person and environment involves beliefs and cognitive competencies developed and modified by social influences.

The interaction between the environment and their behaviours involves the person's behaviour determining their environment, which in turn affects their behaviour [15].

This SCT has got reputation because it explains individual acts specifically point (i) above. More precisely, the

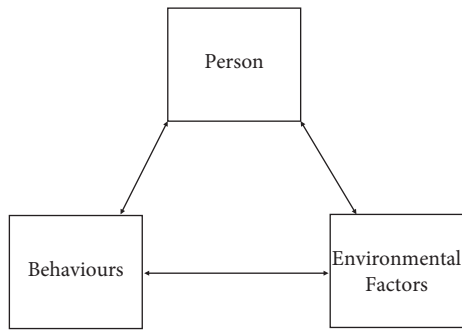


FIGURE 1: Social cognitive theory (SCT).

SCT argue that the person's cognition like self-assessment can guide the behaviour without the actual skills an individual has [4]. For example, the authors have defined a social cognitive conceptual framework using SCT, to simplify theory integration in the field of presenteeism research [16].

Metacognitive knowledge has been described as the knowledge and a deeper understanding of the cognitive process and product [6]. For example, a programmer knows that they must test the function that has a specific task but does not need to test a simply declared variable. Metacognitive knowledge has three important components. Declarative metacognitive knowledge was found to be "what is known in a propositional manner" [17] or the assertion about the world and the knowledge of the influencing factors of human thinking. Procedural metacognitive knowledge can be described as the awareness of processes of thinking [17] or the knowledge of the methods for achieving goals and the knowledge of how skills work and how they are to be applied. Conditional or strategic metacognitive knowledge is "the awareness of the conditions that influence learning such as why strategies are effective, when they should be applied, and when they are appropriate" [17].

2.2. Metacognitive Monitoring. Metacognitive monitoring means the ability to successfully judge one's own cognitive process [7]. According to the author in [9], "metacognitive monitoring is the conscious monitoring, control, and active regulation of the self-cognitive activities in the whole process of cognitive activity." The author argues that metacognitive monitoring is self-monitoring for which the individual must have to require some knowledge and experience of metacognition. The real taste of metacognition can be felt in metacognitive monitoring. Metacognitive monitoring affects the regulation of study, which in turn affects overall learning [18].

2.3. Metacognitive Control. Metacognitive control can be defined as the ability to regulate cognitive activity. Metacognitive control is used to control the memory process and to use strategy to improve comprehension. By using a learning strategy, the learner gains knowledge in the area of concentration. The learner will learn to ask themselves whether they are looking for directly stated, implied, or need to connect information from their own backgrounds with

the material [17]. Recently, the author in [19] found that due to cognitive control training, the key neural region has been improved in terms of active and proactive activities. To understand the conceptual mapping of different learning strategies, we have represented these different learning strategies in Figure 2.

2.4. Self-Assessment. Self-assessment is defined by the author in [20] as "Self-assessment is a process of formative assessment during which students reflect on and evaluate the quality of their work and their learning, judge the degree to which they reflect explicitly stated goals or criteria, identify strengths and weaknesses in their work, and revise accordingly" (2007, p.160). Investigating the present abilities of oneself is the basic step to learning. Self-assessment builds a way to check out the progress of oneself, after that, one can understand the further improvement in the learning process. It can motivate the learner about his/her responsibility and work. It makes the learner possible to focus on improvement by awarding him/her about their cognition abilities.

Self-assessment abilities commonly can repair what students do not see as superior to anything educators can, in light of the fact that instructors for the most part cannot analyze as absolutely the student comprehension problem [21]. Mostly, learners do not assess themselves until they are guided or made aware to do so [22]. Hence, it is important to assess those individuals so that they can use their self-assessment abilities to solve a problem in any situation.

The authors in [23] argue that "the way in which self-assessment is implemented is critical to its acceptance by students". Different techniques have been used to assess the self-assessment of different types of learners. For example, the authors in [21] use ACE in ITS to assess the student self-exploration on run time, Behlau et al. [24] used questionnaires to find the impact of voice problems on individual life, Binali et al. [4] use PSE to assess the student of programming, Boud [23] in their study used State University data to measure self-assessment ads Questionnaire Tutors and to assess the self-assessment of learners, also the Self-Assessment Manikin [3, 25] used ELM-PE to measure the self-assessment of a student in the ITS system [8], professional judgment and expertise oriented framework have been developed by the authors in [26] by postulating on the aspirant/developing coach's capacity for and development of metacognition within the reflective process, and a lot more.

2.5. Aspects of Self-Assessment. To promote the metacognition aspects of novice developers working in the software industry, we focused on the three well-known aspects of self-assessment skills [10].

2.5.1. Planning. Learners plan better and learn when their attention focuses on learning objectives [27]. A clear discussion of the learning goals starts the metacognitive process by prioritizing the significance of thinking about the learning process over the content. The authors [28] in their study stated that goal setting and strategic planning

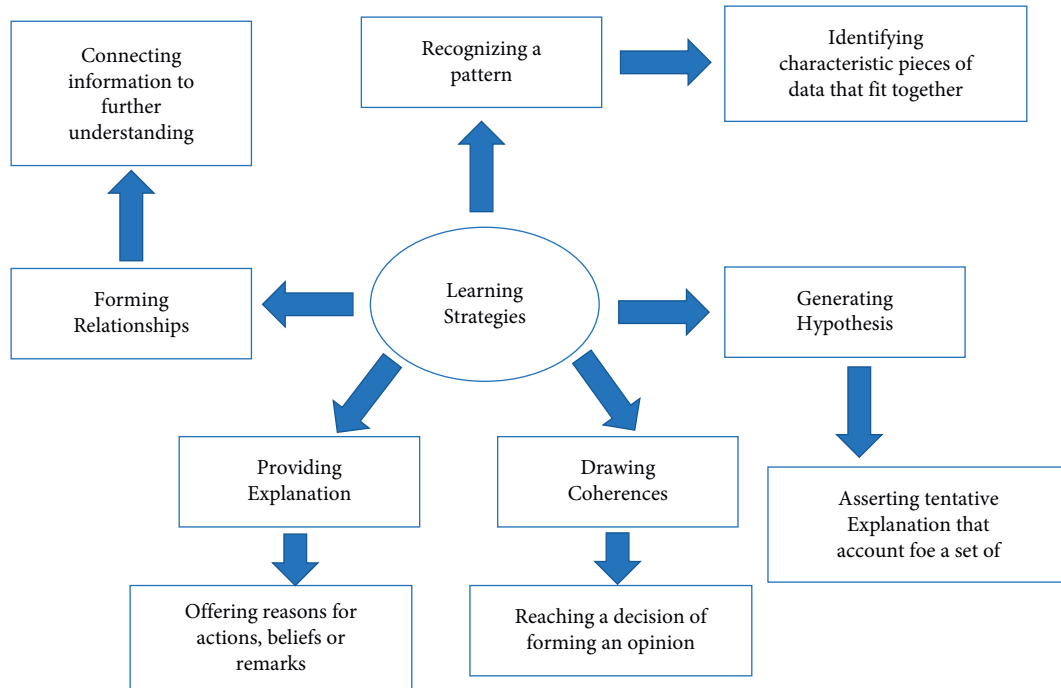


FIGURE 2: Learning strategies.

positively anticipate objective fulfilment in Massive Open Online Course [28]. It is important for every learner to review what is already known about the topic or task on which he/she is working [29]. Also, it is important for the learner to assess the time it will take to complete this task and also to have a clear idea of the resources needed for the successful completion of that task to help him/her think about the process of learning.

While solving a problem, we can divide the type of learner into different categories:

- (i) Learner with initial planning;
- (ii) Learner with continuous planning or partial planning.
- (iii) Learner with no planning

2.5.2. Monitoring. Every learner benefits from monitoring their understanding during learning activities [2]. Grainger et al. [30] also originated that impaired metacognitive monitoring in developmental disorders in children has important educational implications. The importance of the learning process can be reminded by monitoring learning behaviors throughout the learning process. This can be done by chunking, connecting, elaborating, and organizing the materials in such a way that the learner can recognize patterns and associations [31]. In the monitoring process during the work/problem solving, it is easy to figure out the errors and correct them in the thinking process.

2.5.3. Evaluating. By evaluating self-assessment skills, the learner becomes more aware of the self-assessment process and its direct impact on learning [32]. By evaluating

students, the authors describe and allocate value or merit to the quality of their knowledge outcomes [33]. Different types of evaluation methods such as a checklist, rating scales, and questionnaires can help the learners evaluate their thinking during problem-solving. The study conducted by Sierra and Frodden [34] concluded that those learners who evaluate their learning process can decide whether they want to continue or need to change the learning process.

Students with self-evaluation judgment ability have high-performance feedback from those who do not evaluate their learning process [35].

2.6. Implicit Thinking. Implicit thinking refers to unconscious effects such as knowledge, perception, or memory that impact a person's behaviour even while the individual is unaware of those influences [36]. There are times when we act on something and then consider how we might handle it in a different setting or approach. That is implicit cognition at work; the mind will then proceed based on ethical and comparable scenarios when engaging with a certain notion. Implicit cognition and its automatic cognitive process allow a person to make a decision on the occasion. It is frequently characterized as an automatic process in which actions are easily performed without conscious awareness.

3. Analysis of Self-Assessment Skills

This study followed a qualitative approach where we used observation-based research to observe five novice developers at coding tasks. These developers were also required to fill in self-assessment logs daily for five days.

The design of the research is as follows:

- (i) Define the target participants and select the sampling process.
- (ii) Task preparation
- (iii) Tools used
- (iv) Prepare experiment environment
- (v) Procedure
- (vi) Develop and design self-assessment log
- (vii) Data collection

3.1. Defining the Target Participants and Select the Sampling Process. The populations of our research were novice software developers working in the professional software development industry. The criterion was set to choose those professional developers whose work experience is less than two years. As it was not feasible to analyze the whole population, we selected a convenience sample of five novice developers for experiment and assessment. All these developers were working on Android game development using C# as a development language at a local software firm.

3.2. Task Preparation. We assigned a set of tasks to each developer for five days. The experiment included five tasks, one task for each developer daily, divided into three categories:

- (i) Simple task (ST)
- (ii) Intermediate task (IT)
- (iii) Complex task (CT)

3.3. Tools Used. We selected the Devskiller for observation assessment. Devskiller is a tool that records the time and actions performed by the programmer while performing their coding tasks. Devskiller has been used in other studies for the same purpose [37]. It provides a 14-day trial which was enough for us as our experiment was only five days. On each day, we assigned a specific task in the Devskiller user dashboard and sent invitations to each developer through e-mail. We used Devskiller to investigate different attributes of the developers. These attributes will be discussed in the following section.

3.4. Preparation of Experiment Environment. The five developers were provided seating on separate tables in the same room. As the surrounding environment is also an important factor in self-assessment, the environment was set to be more comfortable and conducive. They were briefed about the purpose of the experiment, the 5-day activities, the plan, and the ultimate goal of the research.

3.5. Procedure. The five tasks that the developers had to complete were divided into three categories. For the first and second days of the experiment, every developer was given a simple task (ST) to complete. The reason to assign simple tasks on days 1 and 2 was to help the programmer revise the

basics of coding. On the third and fourth days, some intermediate-level complex task was assigned. On the last day of the experiment, a complex task was assigned to the developers. Table 1 shows the specific task against each day. Aside from the practical task, the developers also filled out the self-assessment log at the end of the daily task.

3.6. Develop and Design Self-Assessment Log. To complement our findings from the data collected through Devskiller, we used a self-assessment log that the developers had to fill out at the end of the daily task. Figure 3 provides an overview of the questionnaire. The log helped us to make our observations reliable. Both, Devskiller log data and data collected through self-assessment log were used to answer our research questions, which are given below:

- (i) What role can self-assessment play in the lives of novice software developers in minimizing repetitive mistakes?
- (ii) How can self-assessment skills help novice software developers in the field of programming?
- (iii) What is the effect of self-assessment on novice developers' work?
- (iv) What is the difference between developers who use self-assessment and those who do not?
- (v) What kinds of mistakes a novice developer makes without analyzing their work?

3.7. Data Collection. To analyze a developer in terms of self-assessment, data could be obtained in two ways: direct source or indirect source. Directly sourced data comes from the developers' first-hand, which means that the data are collected through the log file of Devskiller and a self-assessment log from developers. Indirect sources are the managerial staff of the software house, where data can be collected through interviews with the team managers and team leads. We used the direct source to collect the data. All the data were collected through the log file of Devskiller and the self-assessment log in Figure 3.

The process of self-assessment includes three main steps, i.e., planning, monitoring, and evaluation. We have considered these three parameters for the self-assessment of developers (see section 3.2.8). In the planning phase, the two other parameters, time taken to make a plan and construct used for planning, were assessed. In the monitoring phase, the number of repetitive mistakes and the type of repetitive mistakes were analyzed and measured, and in the evaluation phase, the technique used by the developer to evaluate his/her solution was analyzed. At the end of the task, the developers were asked to fill in the self-assessment log. The experiment continued for five days.

4. Experiment Evaluation

4.1. Calculating the Attribute of Individual Developers. As discussed, we selected the planning, monitoring, and evaluation phases of self-assessment. A total of seven attributes

TABLE 1: Developers' daily tasks.

Day	Task
1	Write a function that returns the largest element in a list.
2	Write a function that computes the list of the first 100 Fibonacci numbers.
3	Handle the switch statement for the buttons in the first scene for a game.
4	Show the complete panel in the game when an object reaches a specific point.
5	Lock and unlock the levels of the game on the basis of previous level conditions.

Developer Survey Questionnaire	
S.No	Questions
1	What was your task?
Answer	
2	How did you planned your task?
Answer	
3	Did you used internet for help? If yes, what did you used it for?
Answer	
4	While coding, did you make mistakes? If yes, did you solved them?
Answer	
5	Did you completed your task?
Answer	
6	If you completed your task? How did you evaluated you rsolution?
Answer	

FIGURE 3: Self-assessment log.

for these phases were used to assess the self-assessment skills of individual developers. The data about these attributes were collected from the log file of Devskiller and the self-assessment log filled in by the developers on a daily basis. The different attributes that were selected for each phase are shown in Figure 4.

The data about the planning phase were collected from two attributes.

- (i) The time taken by the developer to plan for the task
- (ii) The construct used by the developer for planning. This included any pen and paper planning and workflow designs.

The number of repetitive mistakes done by the developer was gathered from the Devskiller log. The log data stores information about all the actions performed by the developer including typos and edits.

The evaluation phase was analyzed by the strategy used by the developer to evaluate his solution. We categorized these strategies into three types.

- (i) Run code directly; the developer did not spend time in evaluating his code before trying to run it.
- (ii) Dry run: the developer did a dry run of their code to ensure it would work as required.
- (iii) Used test cases: the developer generated test cases to test his solution for completeness.

4.2. Assessing Developers Based on Attributes. The self-assessment log was designed to collect information regarding the knowledge and ability of self-assessment of developers in terms of planning, monitoring, and evaluation. Table 2 shows an overview of the data collected from the self-assessment log.

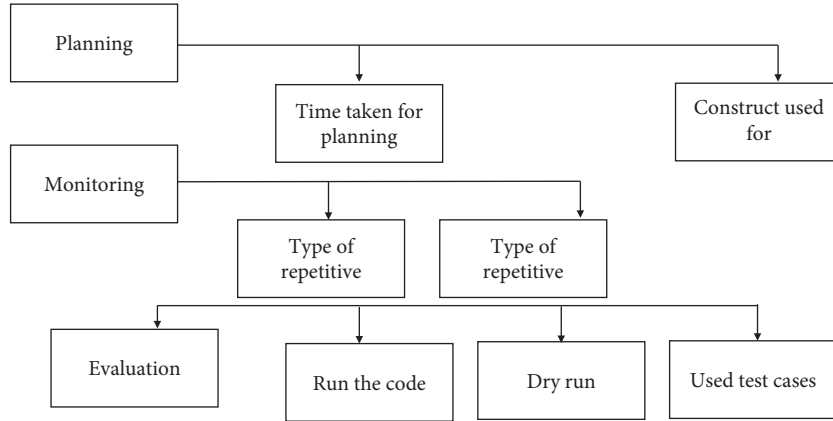


FIGURE 4: Selected attributes for self-assessment phases.

TABLE 2: Developer planning, monitoring, and evaluation data from the assessment log.

Developer	Days	Planning		Evaluation	
		Planned or not	Construct used for planning	Evaluate solution or not	The method used for evaluation
1	1	No	Search Internet	Not	Run
	2	No	Search Internet	Not	Run
	3	No	Search Internet	Not	Run
	4	Yes	Implicit thinking	Yes	Dry run
	5	Yes	Implicit thinking	Yes	Dry run

TABLE 3: Developer planning, monitoring, and evaluation data from the Devskiller log.

Developer	Day	Planning	Monitoring	Evaluation
		Time taken for planning	Number of repetitive mistakes	Task completed?
1	1	0	15	No
	2	0	14	No
	3	0	13	No
	4	1 minute	12	Yes
	5	2 minutes	8	Yes

TABLE 4: Developer planning, monitoring, and evaluation data.

Developer	Days	Planning			Monitoring	Solution evaluated before running	Evaluation	
		Plan made for the task	Time taken for planning	Construct used for planning			The method used for evaluation	Task completed
1	1	No	0	Search Internet	15	No	Run	No
	2	No	0	Search Internet	14	No	Run	No
	3	No	0	Search Internet	13	No	Run	No
	4	Yes	1 minute	Implicit thinking	12	Yes	Dry run	Yes
	5	Yes	2 minutes	Implicit thinking	8	Yes	Dry run	Yes

Along with the data collected from the self-assessment log, we also collected data from the Devskiller log file. Table 3 shows an overview of the data collected from the Devskiller log file.

Collectively, the data gathered from the assessment log and Devskiller log file is presented in Table 4.

Table 4 shows the use of self-assessment skills of developer 1 for five days from three perspectives. The first

subcolumn of the planning phase tells us that if the developer made any plan to solve the task assigned. If the developer made any plan, the value will be YES, and if they did not make any plan the value will be NO. The next column shows the time taken by the developer for planning. If no plan was made, then the value will be 0. The next column of the planning phase tells us about the strategy used by the developer for making their plan. The specific value was

TABLE 5: All developers planning, monitoring, and evaluation data.

Dev	Days	Planning		Monitoring		Evaluation	
		Plan made for the task	Time taken for planning	Construct used for planning	Number of repetitive mistakes	Solution evaluated before running	The method used for evaluation
2	1	0	0	Search Internet	10	No	Run
	2	1	2 minutes	Implicit thinking	5	Yes	Dry run
	3	1	3 minutes	Mind mapping	3	Yes	Dry run
	4	1	5 minutes	Mind mapping	3	Yes	Used test cases
	5	1	1 minute	Mind mapping	2	Yes	Used test cases
3	1	No	0	Search Internet	15	Not	Run
	2	No	0	Search Internet	14	Not	Run
	3	Yes	2 minutes	Implicit thinking	8	Yes	Dry run
	4	Yes	3 minutes	Implicit thinking	5	Yes	Dry run
	5	Yes	2 minutes	Mind mapping	2	Yes	Used test cases
4	1	No	0	Search Internet	15	Not	Run
	2	Yes	1 minute	Implicit thinking	8	Not	Run
	3	Yes	2 minutes	Implicit thinking	7	Yes	Dry run
	4	Yes	3 minutes	Thinking	3	Yes	Dry run
	5	Yes	2 minutes	Mind mapping	2	Yes	Used test cases
5	1	Yes	2 minutes	Implicit thinking	5	Not	Dry run
	2	Yes	2 minutes	Implicit thinking	4	Not	Dry run
	3	Yes	3 minutes	Implicit thinking	2	Yes	Dry run
	4	Yes	2 minutes	Mind mapping	2	Yes	Used test cases
	5	Yes	4 minutes	Mind mapping	1	Yes	Used test cases

populated using the strategy used by the developer. The next column monitoring has one subcolumn “Number of repetitive mistakes.” This column shows the number of repetitive mistakes made by the developer. The value of this parameter was retrieved from the Devskiller log file. For the evaluation phase, we checked whether the developer evaluated their code before trying to run it. If they did, the next column specifies the method used by the developer to evaluate his solution. The last column task completed or not shows the status of the task given to the developer on the corresponding day. All the collected data from five developers are presented in Table 5.

4.3. Effect of Self-Assessment on Task Completion. From Table 5 and the self-assessment log, we generated Table 6 to show the effect of self-assessment on the task completion of the developers. For self-assessment, it is important that the individual uses his self-assessment abilities uniformly in all the important phases of assessment such as planning, monitoring, and evaluation. The table shows the values of self-assessment and status of task of Developer 1 for five days.

Table 6 shows a positive correlation between the use of self-assessment skills and task completion. The data for the rest of the developers are presented in Table 7.

From the data in Table 7, it is evident that there is always a positive correlation between the use of self-assessment and task completion status. However, there was an exception in the case of developer 3 where on the first day, he did not use any self-assessment ability but still completed the task. On further investigation, it was revealed that the developer had actually used an Internet source and copy-pasted the code that he had found online to complete his task. This does not

TABLE 6: Effect of self-assessment on task completion.

Days	Self-assessment	Task status
1	No	Not completed
2	No	Not completed
3	No	Not completed
4	Yes	Completed
5	Yes	Completed

TABLE 7: Effect of self-assessment on task completion.

Developer	Day	SA	Task status
2	1	No	Not completed
	2	Yes	Completed
	3	Yes	Completed
	4	Yes	Completed
	5	Yes	Completed
3	1	No	Completed
	2	No	Not completed
	3	Yes	Not completed
	4	Yes	Completed
	5	Yes	Completed
4	1	No	Not completed
	2	No	Not completed
	3	Yes	Completed
	4	Yes	Completed
	5	Yes	Completed
5	1	Yes	Completed
	2	Yes	Completed
	3	Yes	Completed
	4	Yes	Completed
	5	Yes	Completed

stand true for all situations in the professional industry since there are many situations for which readily available code is

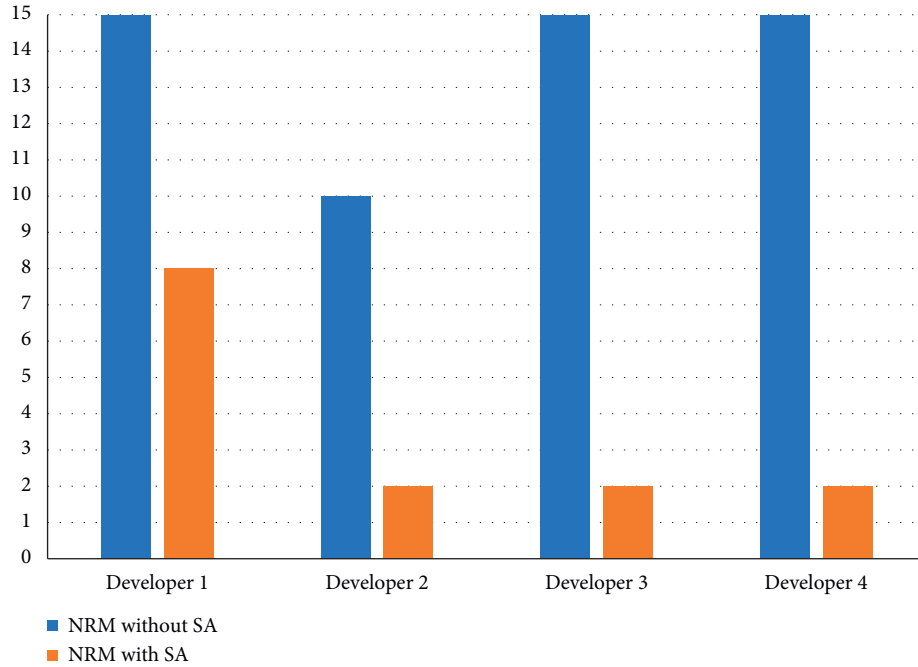


FIGURE 5: Comparison of NRM with SA status.

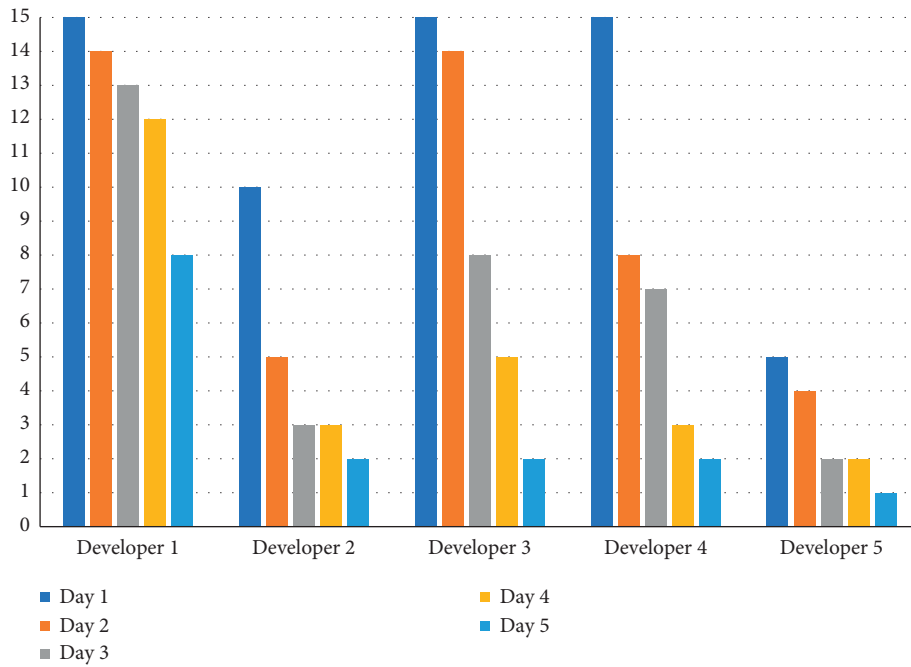


FIGURE 6: Number of repetitive mistakes of five developers.

not available and the developer needs to use their experience and skills to solve such problems.

4.4. Effect of Self-Assessment on Repetitive Mistakes. The overall effect of self-assessment on repetitive mistakes is reported in Figure 5.

Figure 6 shows the effects of self-assessment on repetitive mistakes. From the results, it can be analyzed that the use of

self-assessment skills is negatively correlated to the number of repetitive mistakes. Taking the case of developer 2, on the first day, he did not assess himself and made 10 repetitive mistakes. Since he was more informed on the second day, he was careful to assess himself and be vigilant about what was expected of him. As a result, the number of mistakes he made during the coding exercise decreased to 5. Developer 1's performance showed the least effect of using self-assessment skills on task performance. His number of mistakes

did not see a very big change; however, this number continued to decrease with every passing day, which was a positive sign. This result showed that while still not assessing himself thoroughly, there was still some improvement in his performance due to the awareness of self-assessment. This could also have been affected by the notion of the experiment and therefore would require further investigation which was not the scope of this study.

4.5. Types of Repetitive Mistakes. From the programming perspective, we analyzed only three types of mistakes which were frequent to occur. These were

- (i) Syntax errors
- (ii) Semantic errors
- (iii) Logical errors

The complete data about these mistakes made by developers are shown in Table 8.

For the purpose of analysis, we compared the data on the number of repetitive mistakes (NRMs) for each developer from day 1 and day 5. This was done to investigate whether the experiment had any effect on the use of self-assessment skills by developers and whether this affected the frequency of repetitive mistakes. Table 9 shows the number of mistakes repeated by developers on both day 1 and day 5, respectively.

Figure 5 visualizes the data of Table 9 with categories: NRM (number of repetitive mistakes) without using SA (self-assessment) and NRM with SA.

5. Results and Findings

From the above-performed experiments, we tried to find answers to the following questions.

5.1. What Role Can Self-Assessment Play in the Lives of Novice Software Developers in Minimizing Repetitive Mistakes? The analysis of the results from the experiment conducted revealed a negative correlation between the use of self-assessment abilities and the number of repetitive mistakes. The results showed a clear diminishment in the number of mistakes made by the developers as the experiment progressed. One of the major factors influencing this trend was the awareness of the intent of the experiment which implicitly led the developers into engaging self-assessment activities. The results provide a clear insight that developers who used self-assessment made fewer repetitive mistakes compared to those who did not assess their work. It is also clear from the results that the awareness of self-assessment has a greater impact on the developer's aptitude.

5.2. What Kinds of Mistakes a Novice Developer Makes without Analyzing Their Work? There are three types of mistakes that every developer makes repetitively in everyday work. These mistakes along with their repetition times are presented in Table 9 and visualized in Figure 5. Syntactic mistakes have more frequency compared to other types of mistakes. Syntax mistakes are mostly related to the

TABLE 8: Number of repetitive mistakes of 5 developers.

Developer	Day	Syntax	Semantic	Logical
1	1	8	4	3
	2	7	5	2
	3	8	3	2
	4	9	2	1
	5	3	3	2
2	1	6	2	2
	2	3	1	1
	3	3	0	0
	4	2	0	1
	5	2	0	0
3	1	8	4	3
	2	9	3	2
	3	5	2	1
	4	4	1	0
	5	2	0	0
4	1	7	5	3
	2	4	2	2
	3	4	2	1
	4	3	0	0
	5	2	0	0
5	1	3	1	1
	2	3	1	0
	3	2	0	0
	4	2	0	0
	5	1	0	0

TABLE 9: Developers' NRMs on days 1 and 5.

Developer	NRMs without SA for day 1			NRMs with SA for day 5		
	Syntax	Semantic	Logical	Syntax	Semantic	Logical
1	8	4	3	3	3	2
2	6	2	2	2	0	0
3	8	4	3	2	0	0
4	7	5	3	2	0	0

knowledge of a certain language. Since all the developers involved in this experiment were fresh graduates, they were expected to have an acceptable level of knowledge of the language syntax. Most of the syntax errors that were seen were due to negligence and the fact that the developer did not monitor their own coding practice. The developers showed haste to finish the task and in doing so made plenty of syntax errors. Another important aspect that was discovered here was the dependence on the Integrated Development Environment (IDE) tool. Most of the developers were unsure about their mistakes because of their experience with IDEs. Since all current IDEs provide the IntelliSense feature which highlights syntax errors, most young developers depend upon such tools to keep track of syntax errors. This affects their performance as this allows them to be careless which is why they do not monitor their own coding while performing tasks.

5.3. What Is the Difference between Developers Who Use Self-Assessment and Those Who Did Not? The experiment was

conducted for five days, and during these five days, we saw a clear pattern emerge in terms of the developers who used their self-assessment skills and those who did not. The self-assessment log provided us with first-hand data about the understanding of self-assessment by each developer. It was noted that the developers who were not sure about their metacognitive abilities provided short answers and were reluctant to share more details about the procedures they followed. In contrast, the developers who were more confident about the use of self-assessment were also able to describe their processes more clearly. The performance evaluation also increased with the experiment since the developers started producing fewer errors in their code because they were continuously monitoring and evaluating their work.

6. Conclusion

The purpose of this research was to evaluate novice professional software developers in terms of planning, monitoring, and evaluation to assess their self-assessment skills to improve their work by minimizing repetitive mistakes they make every day. A thorough literature review, a planned experiment and an assessment log were used to identify the factors affecting the performance of developers and reasons for repetitive mistakes and verify the effectiveness of self-assessment on repetitive mistakes through an experiment.

There is a need for self-assessment awareness in the professional software development industry to enhance the self-assessment skills and work of novice software developers. By applying different methods to analyze the self-assessment skills of novice developers like collecting the log data from Devskiller, use of self-assessment log and static observation of developers during a planned experiment. We identified important factors of self-assessment that can influence the work of young developers. From the Devskiller log, self-assessment log data and results can be extracted that self-assessment has a positive correlation with the performance of novice software developers.

This research also addresses the type of mistakes that novice software developer makes repetitively during coding. From the Devskiller log file and static observation of the experiment, it is concluded that most of the developers make syntactic mistakes repetitively because they do not monitor themselves while coding. One of the main reasons for this was found to be their dependence on the latest IDEs which provide them with the facilities of syntax highlighting syntax errors [38–40].

Data Availability

The data used in this research can be obtained from the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

“The authors are grateful to the Taif University Researchers Supporting Project number (TURSP-2020/36), Taif University, Taif, Saudi Arabia.

References

- [1] D. Emiliano de Souza, C. Favoretto, and M. M. Carvalho, “Knowledge management, absorptive and dynamic capacities and Project success: a review and framework,” *Engineering Management Journal*, vol. 28, pp. 1–20, 2021.
- [2] I. U. Haq, A. Anwar, I. Basharat, and K. Sultan, “Intelligent tutoring supported collaborative learning (itscl): a hybrid framework,” *Learning*, vol. 11, no. 8, 2020.
- [3] S. Aamir and Q. Rifat, “A study of metacognitive knowledge and metacognitive regulation among biology teachers at secondary level,” *Journal of Science Education*, vol. 2, no. 2, 2021.
- [4] T. Binali, C. C. Tsai, and H. Y. Chang, “University students’ profiles of online learning and their relation to online metacognitive regulation and internet-specific epistemic justification,” *Computers & Education*, vol. 175, Article ID 104315, 2021.
- [5] A. Little, J. Tarbox, and K. Alzaabi, “Using acceptance and commitment training to enhance the effectiveness of behavioral skills training,” *Journal of Contextual Behavioral Science*, vol. 16, pp. 9–16, 2020.
- [6] M. Z. Khan, R. Naseem, A. Anwar et al., “A novel approach to automate complex software modularization using a fact extraction system,” *Journal of Mathematics*, vol. 2022, 2022.
- [7] S. Surakka, “What subjects and skills are important for software developers?” *Communications of the ACM*, vol. 50, no. 1, pp. 73–78, 2007.
- [8] P. Brusilovsky, E. Schwarz, and G. Weber, “ELM-ART: an intelligent tutoring system on World Wide Web,” in *Intelligent Tutoring Systems* Springer, Heidelberg, Germany, 1996.
- [9] J. H. Flavell, “Metacognition and cognitive monitoring: a new area of cognitive-developmental inquiry,” *American Psychologist*, vol. 34, no. 10, pp. 906–911, 1979.
- [10] G. Özsoy and A. Ataman, “The effect of metacognitive strategy training on mathematical problem-solving achievement,” *International Electronic Journal of Environmental Education*, vol. 1, no. 2, pp. 67–82, 2017.
- [11] E. Railean, “Metacognition in higher education,” in *Metacognition and Successful Learning Strategies in Higher Education*, pp. 1–21, IGI Global, Pennsylvania, PA, USA, 2017.
- [12] I. Roll, V. Aleven, and Mcl, “Improving students’ help-seeking skills using metacognitive feedback in an intelligent tutoring system,” *Learning and Instruction*, vol. 21, no. 2, pp. 267–280, 2011.
- [13] A. Bandura, *Self-Efficacy, Socialfoundations for Thought and Action: A SocialCognitive Theory*, Prentice-Hall, New Jersey, NJ, USA, 1986.
- [14] J. E. Maddux, “Self-efficacy,” *Interpersonal and Intrapersonal Expectancies*, pp. 55–60, Routledge, England, UK, 2016.
- [15] R. Yilmaz and HafizeKeser, “The impact of interactive environment and metacognitive support on academic achievement and transactional distance in online learning,” *Journal of Educational Computing Research*, vol. 55, no. 1, pp. 95–122, 2017.
- [16] C. Cooper and L. Lu, “Presenteeism as a global phenomenon: unraveling the psychosocial mechanisms from the perspective of social cognitive theory,” *Cross Cultural & Strategic Management*, vol. 23, no. 2, pp. 216–231, 2016.
- [17] E. P. Ross and B. D. Roe, “The case for basic skills programs in higher education,” *Phi Delta Kappa*, Eighth and Union, Box 789, Article ID 47402, Indiana, IN, USA, 1986.
- [18] M. Baars and VD, “Effects of problem solving after worked example study on secondary school children’s monitoring

- accuracy," *Educational Psychology*, vol. 37, no. 7, pp. 810–834, 2017.
- [19] K. D. Vohs and R. F. Baumeister, *Handbook of Self-Regulation: Research, Theory, and Applications*, Guilford Publications, New York, NY, USA, 2016.
 - [20] H. Andrade and Y. Du, "Student responses to criteria-referenced self-assessment," *Assessment & Evaluation in Higher Education*, vol. 32, no. 2, pp. 159–181, 2007.
 - [21] C. Conati and V. Kurt, "Toward computer-based support of meta-cognitive skills: a computational framework to coach self-explanation," *International Journal of Artificial Intelligence in Education*, vol. 11, pp. 389–415, 2000.
 - [22] J. B. Biggs, *Teaching for Quality Learning at university: What the Student Does*, McGraw-Hill Education, Chennai, Tamilnadu, 2011.
 - [23] D. Boud, *Enhancing Learning through Self-Assessment*, Routledge, England, UK, 2013.
 - [24] M. Behlau, G. Madazio, F. Moreti et al., "Efficiency and cutoff values of self-assessment instruments on the impact of a voice problem," *Journal of Voice: Official Journal of the Voice Foundation*, vol. 30, no. 4, pp. 506–e18, 2016.
 - [25] G. McBeath and S. Webb, "Cities, subjectivity and cyberspace," *Imagining Cities: Scripts, Signs, Memory*, pp. 249–260, Routledge, London, UK, 1997.
 - [26] L. Collins, H. J. Carson, and D. Collins, "Metacognition and professional judgment and decision making in coaching: importance, application and evaluation," *International Sport Coaching Journal*, vol. 3, no. 3, pp. 355–361, 2016.
 - [27] I. U. Haq, A. Anwar, I. U. Rehman et al., "Dynamic group formation with intelligent tutor collaborative learning: a novel approach for next generation collaboration," *IEEE Access*, vol. 9, Article ID 143406, 2021.
 - [28] R. F. Kizilcec and Pérez, "Self-regulated learning strategies predict learner behavior and goal attainment in Massive Open Online Courses," *Computers & Education*, vol. 104, pp. 18–33, 2017.
 - [29] J. Ehrlinger, A. L. Mitchum, and C. S. Dweck, "Understanding overconfidence: theories of intelligence, preferential attention, and distorted self-assessment," *Journal of Experimental Social Psychology*, vol. 63, pp. 94–100, 2016.
 - [30] C. Grainger, D. M. Williams, and S. E. Lind, "Metacognitive monitoring and control processes in children with autism spectrum disorder: diminished judgement of confidence accuracy," *Consciousness and Cognition*, vol. 42, pp. 65–74, 2016.
 - [31] A. Macb, A. Gumley, M. Schwannauer, A. Carcione, R. Fisher, and Mcl, "Metacognition, symptoms and premorbid functioning in a First Episode Psychosis sample," *Comprehensive Psychiatry*, vol. 55, no. 2, pp. 268–273, 2014.
 - [32] L. M. Blaschke, "Heutagogy and lifelong learning: a review of heutagogical practice and self-determined learning," *International Review of Research in Open and Distance Learning*, vol. 13, no. 1, pp. 56–71, 2012.
 - [33] E. Panadero, G. T. L. Brown, and J.-W. Strijbos, "The future of student self-assessment: a review of known unknowns and potential directions," *Educational Psychology Review*, vol. 28, no. 4, pp. 803–830, 2016.
 - [34] A. M. Sierra and C. Frodden, "Promoting student autonomy through self-assessment and learning strategies," *HOW Journal*, vol. 10, no. 1, pp. 133–166, 2017.
 - [35] S. Stolp and K. M. Zabrocky, "Contributions of metacognitive and self-regulated learning theories to investigations of calibration of comprehension," *International Electronic Journal of Environmental Education*, vol. 2, no. 1, pp. 7–31, 2017.
 - [36] S. Braun, S. Stegmann, A. S. Hernandez Bark, N. M. Junker, and R. van Dick, "Think manager-think male, think follower-think female: g," *Journal of Applied Social Psychology*, vol. 47, no. 7, pp. 377–388, 2017.
 - [37] 2015, <https://devskiller.com/37-best-articles-from-2015-on-recruiting-programmers-and-tech-talents/>.
 - [38] J. E. Jacobs and G. P. Scott, "Children's metacognition about reading: issues in definition, measurement, and instruction," *Educational Psychologist*, vol. 22, no. 3-4, pp. 255–278, 1987.
 - [39] D. Mukherjee and T. Singh, "Effects of metacognitive awareness, self-efficacy and goal orientations on programming performance of software professionals," *Recent Advances in Psychology: An International Journal*, vol. 3, p. 116, 2016.
 - [40] A. Desoete, "Mathematics and metacognition in adolescents and adults with learning disabilities," *International Electronic Journal of Environmental Education*, vol. 2, no. 1, pp. 82–100, 2017.