Journal of Theoretical and Applied Information Technology
15th September 2020. Vol.98. No 17

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

THE EFFECT OF COGNITIVE FACTORS IN DETERMINING STUDENT'S SUCCESS IN COMPUTER PROGRAMMING

MURIMO BETHEL MUTANGA
Department of Information and Communication Technology, Mangosuthu University of Technology,
Umlazi, Durban, South Africa
mutangamb@mut.ac.za

ABSTRACT

There is a growing reliance on technology as the core driver of the 4th industrial revolution. This trend not only delineates Information Technology (IT) as a key topic of global discussion but also makes programming the most rapidly growing skills required by employers. Also, on the academic front, it challenges the capability of current curricula to produce competent IT graduates armed with the right skill-set to meet the surging demand for IT professionals. Moreover, addressing this challenge goes beyond designing a university curriculum for fields that offer IT courses with a computer programming component because unlike other subjects, students often have little to no experience with computer programming before arriving at the university. Consequently, teaching and learning computer programming becomes more challenging than other subjects, and aside from the direct result in poor students' academic performance, fewer students also master the skill. Generally, the debate on improving student's academic performance has inspired a myriad of investigations into factors with correlative impact. However, while literature significantly links student's academic performance to the impact of cognatic factors, there is still a need to investigate the impact of cognition on subjects. Such investigation has the potential to contribute toward enhancing curriculum development and inform approaches to teaching and learning. Therefore, in this paper, we investigated the effect of cognitive factors on students' performance in introductory programming. Using a case study of undergraduate students at a South African University of Technology, our findings show that enhancing cognitive abilities leads to greater performance in introductory programming. More so, personal motivation was found to be the core driving force behind developing and enhancing cognitive ability.

Keywords: ***Cognitive factors, Cognitive performance, Programming, Curriculum, Learning***

1. INTRODUCTION

The world is moving into the fourth industrial revolution - a new developmental period that has the prospect to converge the application of technology and create a significant and multidimensional influence on every field of human life [1], [2]. This technological revolution will deepen the reliance on everyday operations on technology, as technology becomes more and more pervasive, leading to a paradigm shift from a hardware-centered to software-centered technology [2], thereby making digital technology the pivot and driver of global innovation.

Whether for the student, researcher, or industrialist, this massive transition, promises unlimited prospects in the emergent fields of Internet of Things, cloud computing, artificial intelligence, and machine learning and big data. From a labour market, it means there will be an ever-growing global demand for IT skilled and hands-on personnel necessary for steering the resultant digital economic, social, and other innovative systems in both developed and developing countries [3], [4]. The implication is that computer programming, which is the core of IT skills will become almost indispensable. Therefore, it becomes highly imperative for universities that offer IT courses to strengthen their capability to produce graduates with sound programming skills.

While the demand for graduates with programming skills is on the rise, studies indicate that learning programming is more challenging as evident the reported poor performance of students in programming courses, especially for beginners [5]. For instance, it has been alluded in [6] that it is often a challenging task for many students attending a computer programming course for the first time

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

and that it becomes evident as programming courses often have a significant number of students who either fail or dropout. Moreover, in South Africa, the performance of undergraduate students, in general, has been a concern not only to the instructors and administrators but the country as a whole [7], [8]. This predicament has affected both, private and public sectors of the economy. The students are important assets of the country's economy as they transfer skills from the Universities to the industry. The government spends a lot of money every year to fund universities across the country, hence university throughput is of paramount importance for the government. One area that has seen a low pass rate is the science discipline [9]. The country relies on science and technology graduates to take the country forward.

In this era of digital communications, the importance of IT graduates cannot be underestimated. There is an increasing demand for network engineers, programmers, and other IT-related specialists, yet within South African Universities, the performance of IT students has been of concern to many academics even as the 2017 graduate data shows that the number of undergraduate students that completed a Computer and Information Science degree grew less by 200 compared to 2014. This concern has been corroborated by the Institute of Information Technology Professionals South Africa (IITPSA), which claims that the country's education system is not geared to deliver high-end ICT skills at the scale needed.

This challenge understandably stems from the concern raised in several studies that most students find it difficult to understand introductory programming courses, which leads to their poor performance in it [10]. Towards understanding this challenge, different studies have investigated the underlying factors such as cognitive ability, personality traits, gender, university integration, motivation, family factors, previous programming experience, and self-efficacy, mental model and school previous academic results [11], [12], [13]. These studies report that prior performance, mathematics ability, and previous programming experience have been frequently reported to have a positive effect on academic performance. However, with the world-wide average success rates in introductory programming courses estimated to be 67 percent [14], it becomes highly imperative to investigate 1) How does cognitive ability impact students' performance in introductory programming? 2) Which cognitive factor impacts students' performance in introductory programming most? 3) How does cognitive performance factors affect students' performance in introductory programming?

In contributing to answering these questions, therefore, we investigated the perception of students on the effect of cognitive factors as a predictor of their academic performance in introductory programming. We illustrate these predictive factors through a case study involving two groups of year-2 Information Technology students at a South African University of Technology. The discussion is then framed based upon the cognitive learning theory (CLT) [15] which theory maintains that "knowledge is something that is actively constructed by learners based on their existing cognitive structures".

In exploring how cognitive factors contribute to students' proficient learning of introductory programming, we aim to contribute to the enhancement of curriculum development for South African Universities of Technologies, and toward providing useful insights for crafting intervention programs to assist at-risk students.

In the rest of the paper, we examine the literature on solution approaches to poor performance in introductory programming. We then explored cognitivism as a learning theory and presented our case study design and instrument. Next, the findings of the study are presented, and the paper is summarized and concluded.

2. LOW PASS RATE IN INTRODUCTORY PROGRAMMING: A SOLUTION SPACE ANALYSIS

The phenomenon of low pass rates in introductory programming is well supported in the literature [14], [16], [17], but while most research efforts have been focused on corroborating existing claims about the actual pass rate [18], there is still scanty evidence on the attempts to explain the phenomenon itself. In this section, we attempt to categorize the works done on this subject to define the context of this study. Based on existing literature, we have identified two major solution approaches adopted in the literature towards addressing the phenomenal problem of low pass rate in introductory programming. These are:

2.1 Learning Experience Approach

Beyond identifying predictive factors in introductory programming, these categories of studies are focused on ways to enhance students' performance by improving their learning

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

experiences. Essentially, the argument is that it is almost impossible to always have all the performance predictors to help guide students' recruitment and placement in computer science and IT courses. Consequently, authors in [5], implemented a sub-goal learning framework for a semester-long introductory programming course to explore its longitudinal effects on students' performance. According to the authors, learning sub-goals consistently improved performance.

In another innovative effort reported in [19], an automated assessment system was designed to provide target feedback to novice programmers. Although the authors admitted the need for further research, they noted that the targeted feedback messages may help to promote conceptual change and facilitate learning programming. Similarly, an empirical study that examines the use of cyberlearning and gamification has been proposed as a strategy to improve students' learning in introductory Computer Programming has also been reported [20].

2.2 Proactive and Preventive Approach

The implication of low performance in programming is not only limited to the direct effect on students' programming skills but extends to include costs on delayed graduation. As such, research effort has been expended towards discovering the factors that may have a significant impact on student success in introductory programming. Such efforts ultimately aim at formulating models to enhance teaching and learning and to provide insight into providing preventive guidance for students' placement into appropriate courses [17].

Studies that aligned with the proactive and preventive approach can be further divided into two sub-categories – the first investigate the effect of personal factors, with the other deal with the effects of teaching approaches.

The first aims at identifying the personal factors that can significantly predict students' performance in introductory programming. One of such factors investigated is students' prior academic background can predict. For example, [17] found a correlation between high performance in introductory programming and students with a strong mathematics background. Extending on this finding, authors in [21], expended the scope of students' academic background to include their Grade 12 Mathematics and English scores. According to the authors, students that had prior programming knowledge and or high scores in Mathematics and English Language are likely to perform better in introductory programming. Therefore, it was emphasized that it is imperative to take educational background into account when developing course materials. In another work as reported in [22], it was maintained that a strong self-efficacy as a personal factor has the potential to significantly boost a student's performance in introductory programming.

The other category of studies that deal with the effect of teaching approaches sought to understand how students' performance is possibly impacted by the kind of teaching approach adopted by facilitators. Toward this goal, the "exercises-only" and "lectures combined with exercises" teaching approaches were examined in [23]. The study asserts that the "exercise-only" teaching approach contributes more significantly toward recording high students' performance in introductory programming. In the same vein, the authors in [21] introduced an interactive programming visualization tutorial tool but reported that the tool did not have the desired positive effect on students' performance in programming. This approach stems from the debate on the value of blended learning as an alternative to conventional teaching [10].

Though there is extensive research on the potential effect of cognitive factors on the academic success of undergraduates in general, we argue the need for a more narrowed perspective as the basis for this study. Therefore, this paper aligns with the proactive and preventive approach, with emphasis on the impact of cognitive factors on students' performance in introductory programming.

3. THEORETICAL UNDERPINNING: COGNITIVISM AND ACADEMIC PERFORMANCE

Cognitivism broadly explains how mental processes are influenced by both external and internal factors to produce learning in an individual. As depicted in Fig 1, the theory asserts that the mental processes of learning occur between stimulus and response.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

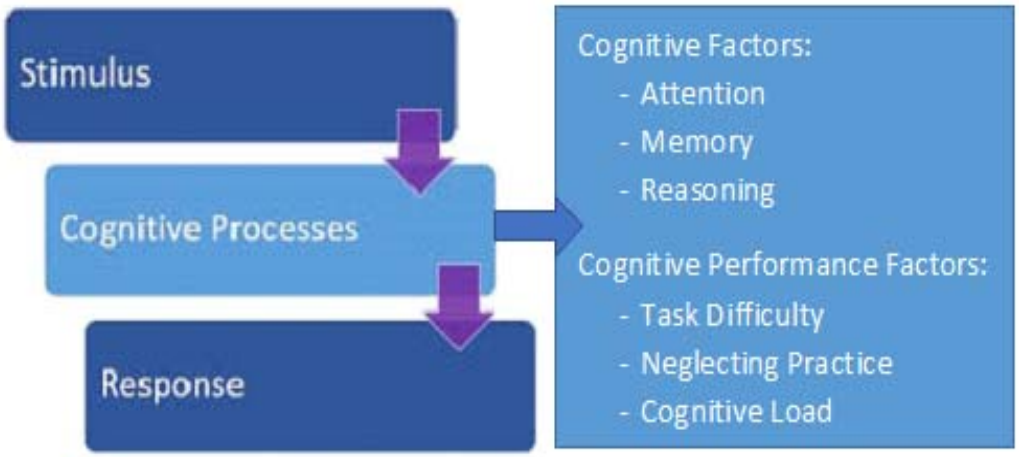

Figure 1: Cognitive learning framework adapted from [25]

It further stresses that the individual takes in the stimulus, processes it in their mind, and then acts upon the stimulus, which alludes to the fact that in the cognitive learning process, new knowledge is built upon prior knowledge. Those processes consist of several elements of the individual which include attention, observation, perception, reasoning, organizing, memory, and forming generalizations. These cognitive elements or factors represent those characteristics of a person that affect the way they learn and perform [24].

Fundamentally, for effective learning, a learner must give thoughtful attention, have a good memory of the relevant information, and be able to carry out independent reasoning of evaluation. Furthermore, to strengthen these cognitive functions, both task difficulty and cognitive load should be low, and the learner must practice whatever is being learned.

We framed our analysis around the three major cognitive factors namely, attention, memory, and reasoning, that play a central role in a programming task. Also, we extend the framework by exploring the contribution of the cognitive performance factors of task difficulty, neglecting practice, and cognitive load to students' performance in programming (illustrated in Figure 1).

4. A CASE STUDY DESIGN AND INSTRUMENTATION

This case study was structured in a manner that allows the researches to explore the effect of cognitive factors on a total of 20 second-year students who were selected for the study. We categorized the study population into two subgroups (subcases) based on the students' first and second semesters average performances in their first year. Each subcase was further partitioned into two equal focus groups. To ensure confidentiality and avoid stigmatization, the basis of the categorization was not disclosed and neither of the two groups knew about the existence of the other. More importantly, we had presumed that participants may not be willing to give out honest information should they know that the investigation equally involves some other students who might know them.

The first subgroup consists of ten students who had an average score of less than 50% in their first and second semester year one introductory programming. The data was officially obtained from the department of information and communication technology by one of the researchers who facilitates the course. Scores within that range depict a poor to fair performance. This categorization condition is meant to enable the researchers to evaluate how cognitive factors may relate to their low performance. On the other hand, the second subcase comprised of another ten students whose average score is 50% or higher. Similarly, this score range of identified students of good to excellent performance.

Because this study investigates a real-world problem, the methodology is designed to ensure flexibility in the process of gaining concrete and in-depth contextual knowledge through two data collection tools – unstructured questionnaires and focus group interviews. Qualitative data may be broader and richer [26], but it may also suffer the deficit of being less precise. Consequently, we used

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

data triangulation and prolonged involvement strategies to enhance precision and strengthen the validity of the study as outlined by Robson [27] and other authors [28]. We achieved data triangulation by using more than one tool to collect the same data on different occasions, which gives the researchers multiple perspectives towards the studied population thereby providing a broader picture. Also, the study leveraged the benefit of prolonged involvement. The long-term relationship that already exists between the participants and researchers, who are both lecturers in Information and Communication Technology Department, enabled the investigators to understand how participants interpret terms used in the study and created an atmosphere of trust that ensured participants spent more time providing data.

Ultimately, the first phase of data collection involved the use of unstructured questionnaires to allow interviewees to articulate their thought unrestricted. We then followed our unstructured questionnaire with open-ended focus groups interview with the participants about their prior programming knowledge, computer literacy, the personal challenge in learning programming, students' attitude in class, access to practice tools, motivation for studying IT, personal study time, study strategies, etc. Some of these questions were raised as a follow up on some interesting views that we had picked up from the completed questionnaires. Adopting the funnel model, the researchers first presented the objectives of the interview and explained how the data from the interview will be used and began by asking open questions that later led to specific questions around cognitive and cognitive performance factors.

To ensure quality and engaging interaction and ease note-taking, we partitioned the study population into four focus groups and interviewed each group separately for 15-30 minutes. By interviewing each interviewee more than once, we aim at gathering data that is both detailed and rich in context. The patterns that emerged from these interviews comprise the bulk of the data

Data collected was first transcribed, coded, and then qualitatively analyzed. As illustrated in Table 1, we designed a two-level coding scheme following Runeson's recommendation [29]: the first-level codes were formulated to allow the researchers to identify and link text in the interview transcripts to the relevant research question that it speaks to. Each first-level code was then further decomposed into second-level codes that allowed each first-level linked text to be linked to a particular construct that it addresses. With this coding system, we developed a case study description of the two subcases.

Table 1: Coding Scheme

Code	Main code	Subcode
1	RqCF: Cognitive Factors	$RqCF_A$: Attention
		$RqCF_M$: Memory
		$RqCF_R$: Reasoning
2	RqCPF: Cognitive Performance Factors	$RqCPF_T$: Task difficulty
		$RqCPF_N$: Neglecting of practice
		$RqCPF_C$: Cognitive load
3	SS_a: Subcase Study 1	Student1_SS_a … Student10_SS_a
4	SS_b: Subcase Study 2	Student1_SS_b … Student10_SS_b
5	RQ	RQ_1
		RQ_2
		RQ_3

5. ANALYSIS OF RESULTS

For each subgroup of the study, we approach the analysis and draw inferences through two sets of observations characterized by a block of related questions that were used to direct the focus of the inquiry. So, in the context of learning programming, we begin by examining how attention, memory, and reasoning as the main aspects of cognitive ability can influence by personal factors such as prior knowledge and skills, amongst others. Another vital component of the analysis is the examination of the modulating effect of the three cognitive performance factors on students' cognitive ability as demonstrated in their performance in introductory programming. was observed.

The overall goal here was to use this set of findings as the basis to explore and explain the relationship between students' performance in introductory programming and their cognitive abilities and

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

cognitive performance factors. With the insight gained from this, we contribute toward the notion of a proactive approach toward addressing the problem of poor performance in introductory programming.

2.3 Analysis For Subgroup Study 1

As our result indicate, cognitive factors are the major determinants of the participants' proficiency in learning programming and account for the quality of their performance. Specifically, participants with weak cognitive ability and failed to strengthen their cognitive function performed poorly in introductory programming. This finding confirms previous works about the impact of cognitive factors on academic performance [30] in general. As earlier explained, the focus of the analysis is this subgroup centres on how the personal factors emanating from the challenges highlighted by the block of interview questions below are linked to the cognitive ability (attention, memory, and reasoning) and cognitive performance factors of participants who scored less than 50% in introductory programming. This choice of the organization follows the pattern learned from the interview transcript and affords and ensured clarity in the discussions.

*A) **Prior Programming Knowledge, Access to Practice Tools/Basic computer Literacy Skills, and Perceived Challenge***

We reported that all ten students who had an average score of less than 50% had no prior programming knowledge before they gained admission into the university. And nine of the participants did not own a personal computer throughout their first-year study. Interestingly, all the students admitted that the inability to access a computer at a convenient time was the major challenge to learn programming during their first year. One student quoted below vividly illustrated the challenge.

Student1_SSa: I didn't have a personal computer and the computer labs were always crowded and my day also occupied with lectures. So for me, getting the opportunity to catch a seat at the computer lab and have sufficient practice time without having to miss other important classes was a monster. Although I could understand the fundamentals, I was unable to do the actual programming.

As explained by Sean Kang [31], "we remember things, because they relate to and can easily be integrated into our existing knowledge base or it's something we retrieve, recount or use repeatedly over time." Based on this, it is evident that the students' lack of both programming background and practice tools, as highlighted above, negatively impacts the cognitive factor of 'memory" and the cognitive performance factor of 'neglecting practice'. This submission aligns with the cognitivist theory assertion that "new knowledge is built upon prior knowledge.". More so, we argue that programming requires a great deal of memory and practice because it is an advanced problem-solving skill that exclusively relies on the application of specific concepts and techniques. Furthermore, as reported in [32], low-ability students perform better in programming when given unlimited computer access to practice.

On computer literacy, only three students reported that they had basic computer literacy skills before commencing university studies. The rest only started learning how to operate a computer at the university. In this, we observed that the open-interview transcripts echo larger students' skill gaps in basic computer literacy and its perceived consequence on their introductory programming proficiency. Though expressed differently as we quote below, they all stressed the same challenge:

Student9_SSa: If I had a chance to learn basic computer skills or programming before my enrolment into first-year, maybe I would have struggled less.

Student4_SSa: "Oh gosh, it was not very good for me at that time. I remember having to spend more time learning how to use the computer than programming itself.

Student10_SSa: It really affected my concentration, especially when we were supposed to do stuff during a lecture. And the worst is that I often fail to pay further attention if I was unable to get the previous exercise – it was frustrating.

All the students acknowledged that the factors listed above affected their ability to understand concepts in programming. However, when asked to rate the factors, programming background and basic computer literacy were rated by seven of the ten students as the factors that mostly influence poor performance in programming.

With these findings, we learn that the lack of basic computer literacy skills can affect the cognitive factor of 'attention'. And according to Veerasamy et al [33], in learning to program, learners are required amongst other things to think,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

understand the concepts of programming, and applied their problem-solving skills. This is where 'attention' comes in handy as a key component of problem-solving skills. And as reported in [34], lack of attention in class is generally is negatively correlated to performance. Concerning the cognitive performance factors, 'task difficulty' and 'cognitive load' were significantly impacted because the students' mental task of learning was now doubled: learning to use the computer and learning programming.

B) Class Attitude, Motivation for Study, and Personal Study Strategies

Implicit in this group of questions is how students internally influence their cognitive and cognitive performance factors. Here we noticed an interesting trend in the data from the questionnaires and transcribed open-interview. First, we had discovered that in responding to a question on how they often felt during programming classes, some participants were very direct with such responses as quoted: "Easily distracted", "I hardly concentrate during the lecture". Second, we also noticed during the open-interview that several students answered "No" to the question of whether they do ask or answer questions in the class. Therefore, to help understand this trend, we decided to introduce a question that would probe their motivation for study.

Interestingly, from this further probe, the following vital insights were gained: five out of the ten students said they were not originally motivated to study IT. Out of these five students, three of them explained that they found themselves studying the course because it was the only one still open as at the time they applied for admission into the institution. In contrast, as quoted below, it was found that the remaining two students were demotivated because they had had a wrong perception of the course. According to one of these two participants coded as Student7_SSa:

I had thought IT was just theory and the kind of practical that would involve how to use the internet. So, in my first programming class, I was like "I'm in the wrong place". That feeling hunted me throughout, and it affected my attention in the class.

The finding relates to the fact that motivation is a dominant driving force in the pursuit of career and other goals because it has a dramatic impact on what a person pays attention to [33]. Moreover, attention as stated earlier is a critical component of cognitive learning. This explains why these students had a poor performance in introductory programming. That is, their lack of motivation reduced their level of attention which in turn, weakened their problem-solving ability.

The aspect of formulating a personal study strategy was variously echoed too. While all the students who were demotivated seemed to not have any precise strategy targeted at learning programming, the others maintained that their study strategy did not work out as expected.

Student1_SSa: I wanted to be practicing as often as I can but that wasn't possible because I had no personal computer. And whenever I had free time at school, the lab computers are either occupied or the place is too rowdy.
Student2_SSa: I had to be staying late at school to use the lab when most students would have gone. However, I didn't still get enough practice time as I wish.

Programming requires students to develop some degree of creative thinking [34]. However, such a skill can be difficult to develop under the kind of scenario described in the experiences shared above. Instead, from the thoughts echoed about personal study strategy, we can infer that the circumstances under which the students learned programming harmed their cognitive factor of 'reasoning'.

2.4 Analysis For Subgroup Study 2

We will see below that cognitive factors and cognitive performance factors all played a significant role in enhancing students' performance in introductory programming. Ultimately, our results reveal a pattern where students with an average score of 50% and above are associated with the positive impact of cognitive factors and cognitive performance factors.

A) Prior Programming Knowledge, Access to Practice Tools/Computer Literacy, and Perceived Challenge

Of the ten students in this subcase, four explained that learning prior programming was not extremely difficult or new to them because they had completed a short course on IT fundamentals before joining the university. Unlike the participants in the previous subcase, we witnessed that seven of the participants in this subcase had personal computers. Surprisingly, even the remaining three participants who did not have a personal computer also did not

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

consider introductory programming too difficult to learn. Compelled by this unexpected observation, more questions were raised to investigate why these participants could develop such confidence and had performed in programming. However, in our conversation, we understood that they had some leverage over several other students.

Student5_SSb: I lived very close to the campus. That enabled me to mostly stay back at school after lectures till late when most students would have left and the lab if almost free. That was how I could meet with other good students that we practise and solve programming problems together.

From this finding, we can conjecture a link to some factors examined by this study. Staying late to use the lab computers, for instance, is a direct opposite of the 'neglecting practice' cognitive performance factor, therefore, it enhanced the participants' ability to learn to program [31]. Drawing from previous works, we explain that constant practice is highly needed for students to progress in their learning of programming, especially as novices [35]. Also, the reported interaction with other students that are good in programming stands to promote independent learning and enrich the cognitive factor of 'reasoning' [36] which is the strength that gives a programmer the capability to analyze, organize, implement, and evaluate the code outcomes. Furthermore, repeated practice is a form of reinforced learning, which can enhance memory as explained by Sean Kang [31].

Compared to the perceptions of subcase study one, computer literacy was not observed to be a major challenge to almost all the participants in this subcase. Although none of them claimed to have been very proficient in using the computing before, we noticed that their thoughts about how computer literacy skills impacted their cognitive abilities toward learning programming converged toward the importance of having access to practice tools. The most explicit view is quoted below.

Student8_SSb: Not that I was initially good or perfect in using the computer and in programming itself. But because I had basic programming knowledge and also started using a personal laptop before I entered school, at least I knew how to do all the basic things with a system. So, I could easily practise on my own in my room and at the lab, I didn't struggle and was even helping to guide other students who were using the computer for the first time.

Coupled with our earlier report that some participants of this subcase had prior programming knowledge, it can be deduced that the participants' performed well in introductory programming because most of them either had prior programming knowledge or possessed some level of computer literacy skills. In connection with the cognitive factors and cognitive performance factors, we explain that: having prior programming knowledge and possessing basic computer skills helped to decrease participants' cognitive load as reported in [37], enhance memory of the course, and promote their ability to be attentive in class. Having the time to practise also reduces cognitive load and eases the mental difficulty of the task of learning programming.

B) Class Attitude, Motivation for Study, and Personal Study Strategies

One distinctive characteristic of those participants that performed above average in introductory programming was observed in the individual attitude. For example, more than half of them either ask or attempt to answer questions in the class. A particular participant coded as Student3_SSb even employed the strategy of attempting to answer questions in the class to indirectly seek more clarification from the facilitator.

Sometimes when I fail to understand what has been explained in the class and the lecturer throws a question, I quickly raise my hand to attempt. I know that I don't know the right answer, but I do that often so that the lecturer can correct me properly or even explain further.

In connection with that unique attitude, we observed that eight out of the ten participants in this subcase wanted to study IT right from the onset. This finding points to the role of motivation as was revealed during the open-ended interview.

Student9_SSb: In the institute where I did a short IT course, we were taught about what you can do through programming and how valuable a programmer can be. That as a programmer I don't need to look for a job. Right from then, I really wanted to be a programmer.

As we have alluded in the subcase one, this result highlights the fact that participants who are positively motivated are most likely going to be attentive in class thereby performing better. Another interesting finding is that the characteristic attitude of forming a unique or personal study

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

strategy was predominant amongst participants of subcase study 1. These positive attitudes which include staying late at the lab to get a free computer, engaging in self-organized collaborative learning, and attempting to answer questions in the class as a way to seek more clarification are all driven by motivation, and indicatively enhances cognitive learning. For example, regular practice, and collaborative learning, have a positive effect on boosting the three cognitive factors of long-term learning memory, reasoning, and attention. Cognitive performance factors are equally impacted positively by a participant's motivational state because regular practice and collaborative learning jointly helped the participant not to underperform by neglectfully depriving their brain of practice.

6. SUMMARY AND CONCLUSION

Based on the cognitivist framework, this study's instrument was designed to elicit responses on participants' prior programming knowledge, computer literacy skills, access to practice tools, personal motivation, attitudes in class, and perceived challenges in learning programming. The aim is to investigate how cognitive abilities influence participants' performance in introductory programming and how participants can develop or enhance their cognitive functions.

Our findings show that memory, among other cognitive factors, has the strongest effect on participants' performance in introductory programming, which agrees with previous works on the impact of cognitive factors on academic performance in general. This agrees with Sean's [31] work which states reported that reinforced learning or repeated practice over time produces superior long-term learning by promoting the memory of learners.

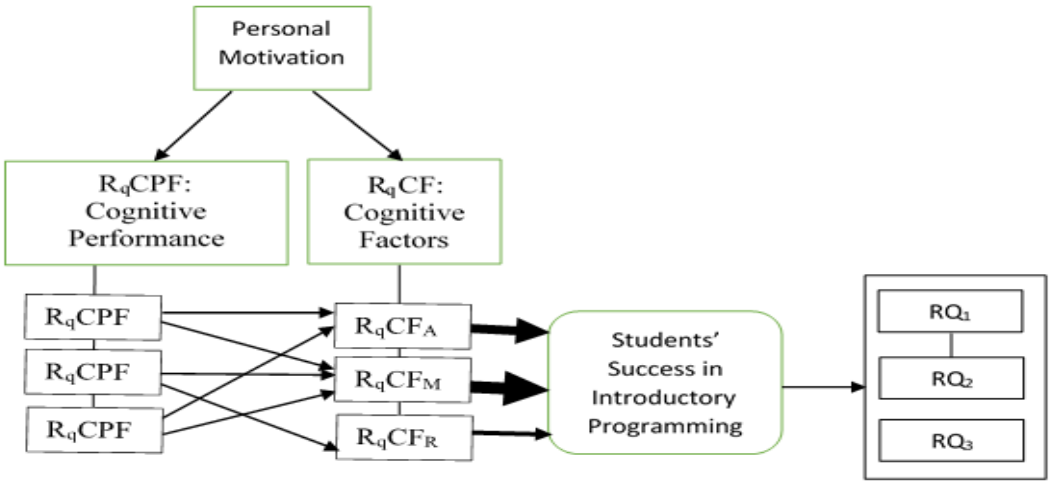

Figure 2: Impact Factor Impact Analysis Model

As illustrated in Figure 2, our findings not only answered the research questions but also offers new insight on how personal motivation acts as an external factor to influence both cognitive factors and cognitive performance factors. Therefore, unlike existing studies which mostly investigated the impact of cognitive ability in general on academic performance, this work uniquely presented an impact factor relationship model (depicted in Figure 2), which shows the determinants of performance in programming as a combination of individual cognitive factors and the and cognitive enhancement factors.

From this model, we drew the key understanding that students' performance in introductory programming is linked to three broad and inter-related factors: cognitive abilities, cognitive performance factors, and personal motivation. Participants with strong cognitive abilities that are supported by positive cognitive enhancement factors were found to have performed better in introductory programming. The current literature also strongly linked performance in computer programming to self-efficacy, individual-interest, positive-thinking, and self-motivation [40]. Furthermore, this study alludes that in as much as cognitive ability might vary with individuals, personal motivation is a vital component that directly and indirectly contributes to developing or enhancing participants' cognitive ability. This postulation resonates with the fact that demotivated participants were associated with a lack of attention in the class and are characterized by poor performance. On the contrary, highly motivated

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

participants were not only found to pay attention in class, they equally demonstrated the unique attitude of formulating personal study strategies such as engagement in self-organized collaborative learning and frequent practice. This again is supported by the finding that attention and motivation are the cornerstones of classroom learning [41]. While the former directly help them to develop the cognitive ability of reasoning, the latter enabled the participants to improve memory because it a cognitive function enhancement factor.

All these sums up to the conclusion that the lack of prior programming knowledge, computer literacy skills, practice tools, and personal motivation can independently impede students' performance in introductory programming. This conclusion summarizes result as demonstrated in Figure 2, which shows that students with strong cognitive factors perform better in introductory programming. And that students who engage in cognitive enhancements activities can indirectly improve their performance in programming by strengthening their cognitive abilities.

This paper, therefore, contributes as follows: first, we contribute toward enhancing teaching and learning by i) scaling the impact of each cognitive ability on the performance of participants in introductory programming and ii) illustrating how cognitive performance factors can increase or decline their cognitive function. This offers useful insight into how facilitators can innovatively organize and their tasks. Second, by highlighting the predominant challenges that impede the development of strong cognitive ability, we contribute both toward enhancing curriculum development and a proactive approach to address performance in introductory programming. Understanding how these challenges interfere with cognition can inform the design of more supportive curricula for new students in a manner the offers them all the leverage to excel in programming.

These contributions provide the basis to suggest that: information technology and basic computer literacy courses should be incorporated into pre-university curricula because these two are the core of the external elements that drive cognitive abilities in the context of learning computer programming.

REFRENCES:

[1] S. Höme, J. Grützner, T. Hadlich, … C. D., and U. 2015, "Semantic industry: challenges for computerized information processing in Industrie 4.0. Automatisierung Stechnik," *degruyter.com*, vol. 63, no. 2, pp. 74–86, 2015.

[2] G. Li, Y. Hou, and A. Wu, "Fourth Industrial Revolution: technological drivers, impacts and coping methods," *Chinese Geogr. Sci.*, vol. 27, no. 4, pp. 626–637, Aug. 2017.

[3] E. Sutherland, "The Fourth Industrial Revolution–The Case of South Africa," *Politikon*, vol. 47, no. 2, pp. 233–252, Apr. 2020.

[4] C. Tsekeris, "Surviving and thriving in the Fourth Industrial Revolution: Digital skills for education and society," *Homo Virtualis*, vol. 2, no. 1, p. 34, Mar. 2019.

[5] L. E. Margulieux, B. B. Morrison, and A. Decker, "Reducing withdrawal and failure rates in introductory programming with subgoal labeled worked examples," *Int. J. STEM Educ.*, vol. 7, no. 1, p. 19, Dec. 2020.

[6] O. Solarte Pabón and L. Machuca Villegas, "Fostering Motivation and Improving Student Performance in an introductory programming course: An Integrated Teaching Approach," *Rev. EIA*, vol. 16, no. 31, p. 65, Jan. 2019.

[7] C. W. Callaghan and E. Papageorgiou, "Personality, Gender and Student Performance at a South African University," *Africa Educ. Rev.*, vol. 17, no. 1, pp. 1–17, Jul. 2020.

[8] M. Reed, M. Maodzwa–Taruvinga, E. S. Ndofirepi, and R. Moosa, "Insights gained from a comparison of South African and Canadian first-generation students: the impact of resilience and resourcefulness on higher education success," *Compare*, vol. 49, no. 6, pp. 964–982, Nov. 2019.

[9] A. A. Ogegbo, E. Gaigher, and T. Salagaram, "Benefits and challenges of lesson study: A case of teaching Physical Sciences in South Africa," *South African J. Educ.*, vol. 39, no. 1, 2019.

[10] A. Alammary, "Blended learning models for introductory programming courses: A systematic review," *PLoS One*, vol. 14, no. 9, p. e0221765, Sep. 2019.

[11] S. Xinogalos, M. Satratzemi, A. Chatzigeorgiou, and D. Tsompanoudi, "Factors Affecting Students' Performance in Distributed Pair Programming," *J. Educ. Comput. Res.*, vol. 57, no. 2, pp. 513–544, Apr. 2019.

[12] G. Kanaparan, R. Cullen, and D. Mason, "Effect of Self-efficacy and Emotional

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Engagement on Introductory Programming Students," *Australas. J. Inf. Syst.*, vol. 23, Jul. 2019.

[13] N. A. Bowman, L. Jarratt, K. C. Culver, and A. M. Segre, "How prior programming experience affects students' pair programming experiences and outcomes," in *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 2019, pp. 170–175.

[14] Simon *et al.*, "Pass rates in introductory programming and in other STEM disciplines," in *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 2019, pp. 53–71.

[15] P. A. Ertmer and T. J. Newby, "Behaviorism, cognitivism, constructivism: Comparing critical features from an instructional design perspective," *Perform. Improv. Q.*, vol. 26, no. 2, pp. 43–71, 2013.

[16] B. Lu, J. Hunt, S. Sudarshan, and J. Fischer, "Examining Strategies to Improve Student Success in CS1," 2019.

[17] E. Tomai and C. F. Reilly, "The impact of math preparedness on introductory programming (CS1) success (abstract only)," in *Proceedings of the 45th ACM technical symposium on Computer science education - SIGCSE '14*, 2014, pp. 711–711.

[18] C. Watson and F. W. B. Li, "Failure rates in introductory programming revisited," in *ITICSE 2014 - Proceedings of the 2014 Innovation and Technology in Computer Science Education Conference*, 2014, pp. 39–44.

[19] Y. Qian and J. D. Lehman, "Using Targeted Feedback to Address Common Student Misconceptions in Introductory Programming: A Data-Driven Approach," *SAGE Open*, vol. 9, no. 4, p. 215824401988513, Jul. 2019.

[20] N. G. Mourya, G. S. Walia, and A. Radermacher, "Using Gamification and Cyber Learning Environment to Improve Students' Learning in an Introductory Computer Programming Course: An Empirical Case Study: American Society for Engineering Education," *ASEE Annual Conference & Exposition*, 2018. [Online]. Available: https://www.asee.org/public/conferences/106/papers/22813/view. [Accessed: 24-May-2020].

[21] M. Schoeman and H. Gelderblom, "The effect of students' educational background and use of a program visualization tool in introductory programming," in *ACM International Conference Proceeding Series*, 2016, pp. 1–10.

[22] M. D. Gurer, I. Cetin, and E. Top, "Factors affecting students' attitudes toward computer programming," *Informatics Educ.*, vol. 18, no. 2, pp. 281–296, Oct. 2019.

[23] X. Zhang, C. Zhang, T. F. Stafford, and P. Zhang, "Teaching Introductory Programming to IS Students: The Impact of Teaching Approaches on Learning Performance," 2013.

[24] J. Sweller, J. J. G. van Merriënboer, and F. Paas, "Cognitive Architecture and Instructional Design: 20 Years Later," *Educational Psychology Review*, vol. 31, no. 2. Springer New York LLC, pp. 261–292, 15-Jun-2019.

[25] PPC, "Cognitivism - The Peak Performance Center," 2020. [Online]. Available: https://thepeakperformancecenter.com/educational-learning/learning/theories/cognitivism/. [Accessed: 26-May-2020].

[26] R. S. Chauhan, "Unstructured interviews: are they really all that bad?," *Hum. Resour. Dev. Int.*, pp. 1–14, 2019.

[27] C. Robson, "Real World Research," in *Real World Research*, 2nd ed., Blackwell Publishing Ltd, 2002.

[28] R. Stake, "The Art of Case Study Research: Data Gathering," *Thousand Oaks, CA Sage*, pp. 49–68, 1995.

[29] P. Runeson, *Case study research in software engineering : guidelines and examples*. Wiley, 2012.

[30] A. Demetriou, S. Kazi, N. Makris, and G. Spanoudis, "Cognitive ability, cognitive self-awareness, and school performance: From childhood to adolescence," *Intelligence*, vol. 79, p. 101432, Mar. 2020.

[31] S. H. K. Kang, "Spaced Repetition Promotes Efficient and Effective Learning," *Policy Insights from Behav. Brain Sci.*, vol. 3, no. 1, pp. 12–19, Mar. 2016.

[32] D. McCormick and S. M. Ross, "Effects of Computer Access and Flowcharting on Students' Attitudes and Performance in Learning Computer Programming," *J. Educ. Comput. Res.*, vol. 6, no. 2, pp. 203–

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

213, May 1990.

[33] A. K. Veerasamy, D. D'Souza, R. Lindén, and M. J. Laakso, "Relationship between perceived problem-solving skills and academic performance of novice learners in introductory programming courses," *J. Comput. Assist. Learn.*, vol. 35, no. 2, pp. 246–255, Apr. 2019.

[34] C. B. Fried, "In-class laptop use and its effects on student learning," *Comput. Educ.*, vol. 50, no. 3, pp. 906–914, Apr. 2008.

[35] R. D. Calcott and E. T. Berkman, "Attentional flexibility during approach and avoidance motivational states: The role of context in shifts of attentional breadth," *J. Exp. Psychol. Gen.*, vol. 143, no. 3, pp. 1393–1408, 2014.

[36] A. Pérez-Poch, N. Olmedo, F. Sánchez, N. Salán, and D. López, "On the influence of creativity in basic programming learning at a first-year Engineering course," *Int. J. Eng. Educ.*, vol. 32, no. 5(B), pp. 2302–2309, Dec. 2016.

[37] P. Silva, E. Costa, and J. R. de Araújo, "An adaptive approach to provide feedback for students in programming problem solving," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019, vol. 11528 LNCS, pp. 14–23.

[38] K. Scager, J. Boonstra, T. Peeters, J. Vulperhorst, and F. Wiegant, "Collaborative learning in higher education: Evoking positive interdependence," *CBE Life Sci. Educ.*, vol. 15, no. 4, p. ar69, Dec. 2016.

[39] L. Zhang, S. Kalyuga, C. Lee, and C. Lei, "Effectiveness of Collaborative Learning of Computer Programming Under Different Learning Group Formations According to Students' Prior Knowledge: A Cognitive Load Perspective," *J. Interact. Learn. Res.*, vol. 27, no. 2, pp. 171–192, May 2016.

[40] I. K. Nti and J. A. Quarcoo, "Self-motivation and Academic Performance In Computer Programming Language Using a Hybridised Machine Learning Technique," *Juanita Ahia Quarcoo Int. J. Artif. Intell. Expert Syst.*, vol. 8, no. 2, pp. 12–30, 2016.

[41] J. C. Horvath, J. M. Lodge, and J. Hattie, *From the Laboratory to the Classroom: Translating Science of Learning for Teachers*, Illustrate. Routledge, Chapman and Hall, 2016.