Journal of Information Technology Education: Innovations in Practice

An Official Publication of the Informing Science Institute
InformingScience.org

JITEiip.org

Volume 20, 2021

TOWARDS UNDERSTANDING INFORMATION SYSTEMS STUDENTS' EXPERIENCE OF LEARNING INTRODUCTORY PROGRAMMING: A PHENOMENOGRAPHIC APPROACH

Irene Govender — University of KwaZulu-Natal, Durban, South Africa — Govenderi4@ukzn.ac.za

ABSTRACT

Aim/Purpose	This study seeks to understand the various ways information systems (IS) students experience introductory programming to inform IS educators on effective pedagogical approaches to teaching programming.
Background	Many students who choose to major in information systems (IS), enter university with little or no experience of learning programming. Few studies have dealt with students' learning to program in the business faculty, who do not necessarily have the computer science goal of programming. It has been shown that undergraduate IS students struggle with programming.
Methodology	The qualitative approach was used in this study to determine students' notions of learning to program and to determine their cognitive processes while learning to program in higher education. A cohort of 47 students, who were majoring in Information Systems within the Bachelor of Commerce degree programme were part of the study. Reflective journals were used to allow students to record their experiences and to study in-depth their insights and experiences of learning to program during the course. Using phenomenographic methods, categories of description that uniquely characterises the various ways IS students experience learning to program were determined.
Contribution	This paper provides educators with empirical evidence on IS students' experiences of learning to program, which play a crucial role in informing IS educators on how they can lend support and modify their pedagogical approach to teach programming to students who do not necessarily need to have the computer science goal of programming. This study contributes additional evidence that suggests more categories of description for IS students within a business degree. It provides valuable pedagogical insights for IS educators, thus contributing to the body of knowledge

Accepting Editor Donna Jean Satterlee | Received: March 17, 2021 | Revised: May 16, 2021 | Accepted: May 18, 2021.
Cite as: Govender, A. (2021). Towards understanding information systems students' experience of learning introductory programming: A phenomenographic approach. *Journal of Information Technology Education: Innovations in Practice, 20,* 81-92. https://doi.org/10.28945/4782

Findings	The findings of this study reveal six ways in which IS students' experience the phenomenon, learning to program. These ways, referred to categories of description, formed an outcome space.
Recommendations for Practitioners	Use the experiences of students identified in this study to determine approach to teaching and tasks or assessments assigned
Recommendations for Researchers	Using phenomenographic methods researchers in IS or IT may determine pedagogical content knowledge in teaching specific aspects of IT or IS.
Impact on Society	More business students would be able to program and improve their logical thinking and coding skills.
Future Research	Implement the recommendations for practice and evaluate the students' performance.
Keywords	information systems, introductory programming, outcome space, phenomenography

INTRODUCTION

It is undoubtedly difficult to learn programming if encountered for the first time at university. Many students who choose to major in information systems (IS), enter university with little or no experience of learning programming. It has been shown that undergraduate IS students struggle with programming (Bashir & Hoque, 2016). This situation is characteristic not only of IS students, but of most students in introductory programming modules (Govender & Grayson, 2007). Programming, which comprises problem solving and coding, involves computational skills. These skills are becoming increasingly important to instil in our students to meet the demands of the 4th industrial revolution (4IR), which encompasses the changes flowing from cutting-edge technology, specifically internet technology, and its affect in steering the progress of how we live, work, and learn. One of the ways of instilling this skill is to teach programming to all if not most students.

Programming is crucial for many reasons, namely, to enable one to innovate, create eco-friendly solutions for global problems, enhance the power of computers, automate, manage, calculate, analyse the processing of data and information accurately, use analytics effectively, and to create software and applications that help computer and mobile users in daily life. Hence, there is no doubt about the importance of learning how to use programming languages in our workspaces. Programming is often the purview of computer science, found in science divisions or colleges. However, Information Systems (IS) is generally in business divisions or colleges. Moreover, programming courses are offered in both science and business programs, and for the most part research done in programming courses is specific to science.

Because of the high failure rate of a module that involves programming and database development, it was determined that an introductory module of programming was needed to scaffold IS students in the field of programming. Most business students did not have the computational thinking skills needed to cope with the changing landscape of business and digital competency. Computational thinking is analogous to a set of problem-solving methods that break a complex problem into smaller problems that can be represented in ways that a computer can execute. Problem-solving is inherently part of programming, hence programming seems to be the vehicle to advance computational thinking. However, as indicated earlier, programming is difficult for many students. Therefore, the aim of this research is to understand IS students' experiences in learning to program with a view to improve the pedagogy and support in teaching programming to IS students. In this paper we determine the experiences of Information Systems students who are learning to program in a business degree programme. Hence the following research question guides this study:

> What are the qualitatively different ways IS students experience learning to program?

The remaining part of the paper proceeds as follows. First, a brief review of the related literature is provided, which is then followed by a description of the methodological framework, Phenomenography. Next, the research methodology is described in detail, followed by the analysis and discussion of results and limitations, and further research is recommended.

LITERATURE REVIEW

It is projected that employment in areas related to computing is set to increase by 13% from 2016 to 2026 (U.S. Department of Labor, 2018), higher than the expected growth for all other jobs. This finding indicates the need for greater digital competence among students. In the present era of fourth Industrial revolution (4IR), a high level of digital competence is demanded of employees. Ferrari (2012) defines digital competence as:

> "the set of knowledge, skills, attitudes, abilities, strategies, and awareness that are required when using ICT [Information and Communications Technology] and digital media to perform tasks; solve problems; communicate; manage information; collaborate; create and share content; and build knowledge effectively, efficiently, appropriately, critically, creatively, autonomously, flexibly, ethically, reflectively for work, leisure, participation, learning, and socializing." (p. 3)

The key aspect of this definition is to solve problems that require computational thinking, which in turn is developed by learning to program. In the business arena competitive decisions are based on the analysis of large and unstructured data. This analysis makes use of machine learning, data analytics and the likes, all of which necessitates knowledge of programming. Scherer et al. (2020) maintains that cultivating skills involved in programming will boost skills in computational thinking. Information systems students have often been on the back foot of programming competence as opposed to their computer science counterparts.

Programming is about solving problems by writing code that is understood by the computer. It has been argued that lack of practice in programming is one of the major reasons for failure in programming (Özmen & Altun, 2014). This view is supported by Niitsoo et al. (2014), who argue that students who spent more time practicing programming during the course performed better academically. While students might develop an algorithm to solve a problem, ultimately the code must be written to achieve a tangible outcome. It is therefore vital to explore Information Systems students' experiences in learning to program.

However, there is consensus among researchers that learning to program is challenging for many students (Abdunabi et al., 2019). Furthermore, several studies have shown high dropout and failure rates in introductory programming courses, which has been a long-standing concern (Dasuki & Quaye, 2016, Vihavainen et al., 2011). The reasons for this dropout rate are many and varied. Of importance is the development of problem-solving skills when learning to program. Moreover, it was determined that students' self-efficacy of programming affects their programming ability (Fasogbon et al., 2016). Therefore, measuring self-efficacy may assist in developing new methods to address the challenges of learning computer programming (Korkmaz & Altun, 2014). Breed et al. (2013), in their study, established that the metacognitive approach to problem solving when programming was beneficial and therefore should be central to teaching and learning. In the same vein, Govender et al. (2014) concluded that "explicit problem-solving instruction" (p. 188) is needed for increasing students' self-efficacy in programming. In a more recent study on novice programmers' misconceptions of programming, it was determined that these lay in the problem-solving plans (Kwon, 2017). Kori et al. (2016) argue that prior exposure to programming increases ICT students' learning motivation and academic achievement.

Research on how students experience programming has been carried out for different cohorts of students using phenomenography (Govender & Grayson, 2007). However, in the reviewed body of lit-

erature no such study has been conducted of business students learning to program using Phenomenography. In this paper I use the framework of Phenomenography to explore and understand IS students experience of the phenomenon of learning to program. In the following section, a brief description of Phenomenography is presented.

PHENOMENOGRAPHY

Phenomenography is a research methodology proposed by Marton (1986) that examines the different ways people experience a phenomenon. In this study the phenomenon is learning to program. Using a phenomenographic approach to understand how people experience learning aspects of a phenomenon has shown to be successful in assisting teachers to adjust their pedagogical approach to teaching the phenomenon.

Phenomenographic research involves related groups of people concerning some phenomena. Phenomenography has been effective in exploring students' notions of learning in different contexts, such as understanding physics concepts (Prosser & Millar, 1989), essay writing (Prosser & Webb, 1994), programming (Booth, 1997), and more recently, mobile learning (Khan et al, 2019). Because students learn and perceive objects in different ways, an array of qualitatively different understandings or experiences of a phenomenon emerge. These different ways can be hierarchically arranged with some capabilities being more involved than others. These differences are, in Marton and Booth's words, "educationally critical" in the learning process (1997, pp.125-126). Also, of note is that the different ways of experiencing the phenomenon are coherently related to one another. This set of ways that emerge from the data is referred to as categories of description in phenomenograhic terms. In other words, phenomenographic methods enables one to assess students' understanding of programming and determine critical aspects of misunderstanding. In this way, a deep rich description of students' understanding of the phenomenon is obtained. Importantly, it enables a holistic understanding by the emerging "different patterns of awareness and non-awareness of component parts" (Åkerlind, 2018, p. 3). The collective set of categories of descriptions then emerge as an outcome space that includes the relationship among the categories. Educators can ask students to talk about a programming concept and audio-record the responses, or they can make use of reflective journals to elicit students' understanding of concepts (Hans & Ellis, 2019).

METHODOLOGY

The study uses qualitative analysis to gain insights into students' experiences of learning programming in a business degree. These students learnt visual Basic as an introductory module in programming. At the time of the study, Visual basic (VB) was the introductory programming language in use in the discipline. VB seemed suitable as a convenient language that makes it fast and easy to create type-safe .NET apps. It is reasonable to assume that this exercise could be applied to any other programming language in use such as, Java, R, Python, C++, and the likes as the learning experiences would be similar in learning to program. The main goal of the study was not to teach a specific language perse but to use the language as a vehicle to learn programming logic. It is claimed by Govender and Grayson (2007) that problem solving skills can be transferred to a new language. Subsequently, the staff changed the language to C#.

DATA COLLECTION

The primary source of data was reflective journals that students were required to keep throughout the course. These journals formed part of the assessment and were submitted electronically in two stages. First, these journals were submitted mid-way through the course, and then the second set was submitted at the end of the course. Specific activities or exercises were needed to be done in the journal. Aside from these activities, students were required to think about their experiences as they worked through the course material. Examples of entries to be written in the journals were thoughts

and ideas, queries and challenges, feelings, summaries, and their experiences of their ongoing learning. Each entry was preceded by the date of the entry. To prevent the idea of "writing for the instructor," there were no right or wrong answers. Hence, marks were given based on how often they wrote in their journals and "how much conscientious thought, honesty and effort went into writing in the journal" (Govender & Grayson, 2007, p. 877). Reflective journals can be more useful to understand the mental processes that students experience as they learn, write, and problem solve (Carr, 2002). Bashan and Holsblat, (2017) concur that data from the journal show what occurs during the learning of any subject or changes in the students' thinking process.

PARTICIPANTS

Two groups of students who were registered for the same introductory programming module within a business degree program participated in the study. The two groups of 31 and 16 students each were lectured to by different lecturers on different campuses. The content and assessments were the same for both groups of students. The questions and comments on problems were based on the text prescribed for the course, programming in Visual Basic. Evidence obtained from the students' journals was enhanced by observations and questioning in class.

ANALYSIS

The journal readings were read by the researcher repeatedly until no more new insights could be found. The first set of journal submissions mid-way through the course were read for common themes and ideas followed by the second set of journal submissions at the end of the course. In reading and understanding the writings of the students, intuition played an important part in the analysis. Common themes related to the learning experiences of students at different times throughout the course emerged.

VALIDITY

To determine the validity of the themes derived, another researcher read through the journals to determine if the themes were relevant and correctly identified the experiences. These themes are what Marton (1986) refers to as categories of description. Thereafter, to verify the categories, an arbitrary set of 10 journals were selected to ensure that views were correctly categorised.

FINDINGS

The analysis of the data yielded qualitatively different ways that students experienced the phenomenon, learning to program. These ways are what Marton (1996) refers to as categories of description: (1) Learning the basics; (2) Incremental learning; (3) Develop an Algorithm; (4) Practice; (5) Seeing tangible outcomes; and (6) problem solving. They describe the qualitative difference in the ways the students experienced learning to program. The categories range from the most restricted interpretations of programming (1) to the most comprehensive view of programming (6). The categories can therefore be ordered hierarchically. This means, for example, that an understanding of learning to program as being about learning the basics or syntax reveals a more restricted experience of learning to program than an emphasis on solving a problem. For each category of description, a detailed description together with elucidatory comments from students' reflective journals is presented below.

CATEGORY 1- LEARNING THE BASICS

Some students' initial experience of learning to program is viewed as learning the basics of programming, that is, the syntax of the language. Aspects such as structure of a program, writing assignment statements, and the appropriate use of semi colons etc. are what students perceive as learning to program. This notion of programming is illustrated in the excerpts from participants' journals below.

> A data type or simply type is a classification identifying one of various types of data, such as real, integer or Boolean, that determines the possible values for that type. I do not have any difficulties with variables. I find it easy to declare variables and use them in calculations. I do not like using variable box diagrams although they are useful, I find them time consuming.
>
> Doing it for the first time was tough because, I had this idea that keep thinking that Programming was for smart students and computer students. For me during classes, I had to listen attentively during although I find it confusing to understand the language of programming.
>
> Understanding syntax of the language was a difficult task as I stated that it is my first time studying how to program applications. The syntax was new to me, but when time went by and I practised more most of the VB language became more understandable

CATEGORY 2 – INCREMENTAL LEARNING

In learning to program for the first time, students realised that it is important to learn the content step by step in a logical sequence, starting with the basics and then building upon that. If they missed one of the lessons, it would set them back. Incremental learning is the way to develop a sound understanding of programming. As one student put it:

> On this day I learned that in this module, there is nothing that has less importance than the other. There is a relation between the chapters. So, it's very important for the student not to forget what they did before.

Another participant explains how she is learning step by step and the importance of scaffolding, as indicated in the following excerpt:

> We started chapter three "Memory locations and calculations" … things are getting serious now we are now coding. …I understand Declaring variables and that you cannot use variables unless they are declared but my problem arises on page 16, example 1 where they "Dim quantity As integer quantity = 650" why don't they just say "Dim quantity As integer = 650" like they did in page 13, example 4 where they declared and initialized in one sentence. The "convert class method" was also tricky at first but I eventually understood it plus the lecturer said our practical will include this method, so we must know it. "Arithmetic expressions" unlike mathematics it seems like BODMAS does not apply in this language, or does it? There are six expressions and they are ordered in precedence number from 1-6, there is an expression called modulus where you divide and the answer is only the Reminder, page 26. We were also asked to code the "Sunshine Cellular application" of which submission day is Tomorrow!!

Yet another participant said:

> I did have a lot of difficulty understanding variables. I then realised that it is like a placeholder, and that made understanding other code so much easier.

CATEGORY 3 – DEVELOP AN ALGORITHM

In this category, students experienced learning to program as how to develop an algorithm in the form of pseudocode or flowchart.

For example, one of the students said that:

> Also learning programming for the first time can be a daunting task, particularly if required to solve a problem.

Another student gave his/her view about an algorithm:

> Yes, I do find it easy to write the steps involved to solve a problem. At first, I found this quite difficult but after learning how to use flowcharts it is much easier to write the pseudo-code out.

Students realised that this is not a normal learning curve – there are different aspects that need to be learnt before finally seeing the outcome.

The comment below illustrates this notion further:

> In working with exercises from the slides and the textbook I found that most of the programs I wrote did not run correctly initially. After consulting with my lecturer and fellow classmates we were able to go through the code and figure out what was wrong with the program. I found that simple errors in my code created incorrect output values.

Another participant alluded to this notion of developing an algorithm.

> We started another chapter which I believe it had brought nightmares to my life, the coding of course. I believe this part is the hardest part, and you should pay much attention. We did some example on the class everything seems so fine.

CATEGORY 4 – PRACTICE

Practicing several examples of a specific concept or structure in programming seems to be the way some students understand and consolidate the concepts. This category of the conception of learning programming is illustrated in the excerpts from several journals below:

> Well to master programming skills I think it is important to keep practicing so by the lecturer giving us activities to do after each chapter, it is helpful and useful. This is how we will overcome the fear that this module is hard. Also, it is important to find the error been made and be careful when coding as you can make silly mistakes like me.
>
> The slides were very helpful because it gave a lot of examples and provided the code, flowchart and pseudocode for the examples provided. I could understand and execute each example perfectly.
>
> … and we even attempted examples in the lecture. This chapter gave me the confidence that I so desperately needed.
>
> The calculations weren't that difficult to master they just took a bit of practice. by this time, I had realized that one must think differently when studying this module as programming is a different language. Studying programming feels like learning a whole new language only understood by a few.
>
> [A]s I learnt the hard way through test one that you need a lot of practice in order to do well.

CATEGORY 5 -SEEING TANGIBLE OUTCOMES

It is important for students to see the product of their learning. The discovery that they can make something appear on the screen as part of the interface is motivating to go further. For example, one participant said:

> I like the way the Splash Screen introduces the application, when I first made the splash screen to appear just before my application open it's like I was really doing something, it's a small thing but to me it was a beginning of something "BIG", at least that how I felt.

This view was echoed by another participant as indicated in the excerpt below:

> Yes, it's nice to see the outcome when running it but truly speaking this is so hard for me. If I still remember clear, this reminds me my first activity that I did on coding, it was about displaying a message using messageBox.Show method. Here's an example … which then when put on run, it's just displayed a small box written my name on it. I was very happy to see this happening and I started to be positive, I started to enjoy doing more activities. All of this I copied it down from slides in the class.

CATEGORY 6 – PROBLEM SOLVING

Problem solving and programming are two sides of the same coin. To learn to program is to learn to problem solve. Ultimately, the goal of writing code is to solve a problem. Some of the participants' excerpts are presented next that relate to the notion of problem solving.

> Yes- It just takes some time to get to an actual solution. I feel personally I must read the question a few times before I can actually grasp what the problem is but after that if you follow a set of steps a solution is easily found.
>
> When solving the problems of strings, I didn't know that I would be faced with an even bigger task of differentiating between an Integer and a Double variable still to this that I still experience some difficulty when encountering these two variables.
>
> Yes, I do find it easy to write the steps involved to solve a problem. I learnt a lot about problem solving in ISTN102 after learning how to use flowcharts and 2IP helped me more by reinforcing what I had already known.
>
> Programming is fun and I enjoy it, it has help me understand computers better and see that most of the applications that we use in the outside are programmed I can now apply practical examples learnt in class to real life issues concerning technology, e.g., interfaces and their controls Facebook, Instagram and twitter etc. all are programmed to set out certain commands when used by users.

DISCUSSION

The purpose of this study is to tease out and understand IS students' experiences of learning to program. Students' notions of learning to program, specifically in information systems as part of a business degree is a valuable concept of investigation in educational research: coding in data science has become mandatory in the business world. The results of this study show that business students' experiences in learning to program can be viewed as six different categories of description. These results may be interpreted in broader contexts as well. The six categories are aligned logically and hierarchically from the lower level to the higher level of understanding, that is, these categories of descriptions build upon each other as indicated in Figure 1. The first two categories of description refer mainly to the first set of journals received mid-way through the course. The remaining four categories emerged mostly from the second set of journal submissions from students. This is in keeping with the trajectory which students experienced learning to program for the first time. It is worth noting that learning to program is a novel way of learning and that practice is vital to master programming. Since programming is synonymous to problem solving, which is the highest level of cognitive ability, the previous steps are important to scaffold to the higher levels of hierarchy. In Category 1, learning the basics is an important experience and can be applied to most other STEM (science, technology, engineering, and math) subjects. It is a category that is experienced in varying time frames for different students. Category 2, incremental learning, is true of most areas of learning but is central to STEM subjects because of the scaffolding that students need to achieve the outcome: in this study the outcome space. Category 3, develop the algorithm, is considered a key step in problem solving, but cannot on its own achieve the output. While we may have a solution in the form of an algorithm,

the actual output requires the code to be written, which requires the syntax or the basics of programming and the language. Other categories (4 and 5) perceived learning to program as a way of enriching student learning regarding computational skills and seeing tangible solutions. Seeing tangible outcomes increases motivation and engagement in learning to program. Ultimately, a problem must be solved. Problem-solving incorporates all the other layers, without which would not attain the goal of programming.

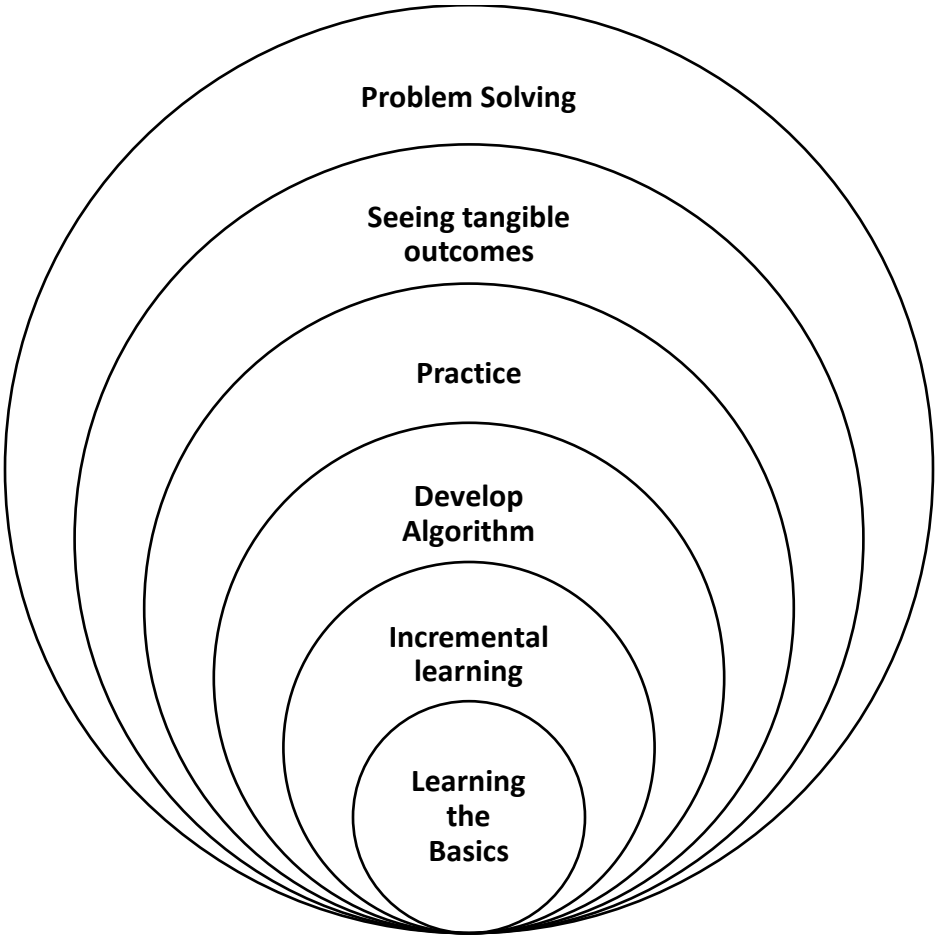

Figure 1: Outcome Space of learning to program for IS students

IMPLICATIONS FOR TEACHING

A key strength of the present study was that participants wrote down their experiences as it was experienced over the duration of the module without compromising the detail. As a result, an in-depth analysis of the data was obtained. They did not have to rely on memory of what happened. Hence, this rich data that informed the findings can be used to develop targeted interventions aimed at reducing the challenges faced by IS students in learning to program linked to each category of experience as suggested below.

Category 1 – Learning the basics. It is crucial that students understand the basics of programming, that is, the syntax of the language and the algorithm of solving the problem. Hence frequent and small assessment tasks should be given to students to overcome the difficulties of the basics of programming.

Category 2 – Incremental learning. Each set of activities or tasks should build upon the previous tasks so that students consolidate their learning and problem-solving skill as they move into the deeper conceptions of learning programming.

Category 3 – Develop an algorithm. To develop an algorithm (a set of steps to reach a solution) is key to developing logical thinking and problem solving. It is worth providing students with a problem and develop a step-by-step algorithm – either using a flowchart or pseudocode first before attempting to code the solution into the programming language. Because seeing the outcome of a piece of code is so exciting for students, it is suggested that very simple instructions can be written initially to show students the output of the code.

Category 4 – Practice. The adage "practice makes perfect" seems to apply in learning, specifically in the context of programming. Practice makes one become familiar with the syntax and steps involved. Instructors are encouraged to give students many practice examples to code with some form of assessment attached to it so that students are compelled to attempt the tasks.

Category 5 – Seeing tangible outcomes. Since students are driven by outcomes and instant gratification, it is necessary to allow students to experience the running of the program often enough to see the outcome. Just using one example program and changing aspects of the code to illustrate the concepts with corresponding changes in output could assist students in learning to program. This category is linked closely with developing the algorithm.

Category 6 – Problem solving is ultimately what students do when learning to program. The development of an algorithm that is experienced in category 3 is fundamental to problem solving. It is advised that instructors provide tasks to solve using the programming language by first clarifying the problem, planning the steps involved, i.e., the algorithm, and then coding the algorithm – if the current programming constructs are not adequate to solve the problem, then introduce a new construct or data structure that will achieve the desired outcome.

CONCLUSION

This study set out to understand the views and experiences of information systems students learning to program with the view to inform IS educators on how they can lend support and modify their pedagogical approach to teach programming to students who do not necessarily have the computer science goal of programming. The study employed phenomenographic methodology to review and examine reflective journal entries of the IS students who participated in the study to discover their thought patterns that might be useful to constructing effective pedagogical strategies for teaching university-level introductory computer programming courses. The main research question was "What are the qualitatively different ways IS students experience learning to program?" The study discovered that business students' experiences in learning computer programming manifested in six logical categories of metacognitive thinking processes and levels of cognitive abilities. In general, the pattern of thought processes and levels of cognitive abilities uncovered can inform the development of approaches that can guide novice problem solvers achieve optimal solutions in problem situations. This research confirms previous findings and contributes additional evidence that suggests more categories of description for Business students learning to program. One cannot prescribe a set of pedagogical practices for different students. However, this study suggests a modified teaching practice as a result of what was discovered in the students' reflective journals. The insights from categories 1 through 6 provide suggestions for teaching practices in each of the categories of learning, thus providing valuable insights for IS educators.

While anecdotal evidence indicates an improved learning curve of business students learning to program, a further study could systematically assess the effects of the intervention as indicated in the section, implications for teaching.

REFERENCES

Abdunabi, R., Hbaci, I., & Ku, H-Y. (2019). Towards enhancing programming self-efficacy perceptions among undergraduate information systems students. *Journal of Information Technology Education: Research*, *18*, 185-210. https://doi.org/10.28945/4308

Åkerlind, G. S. (2018). What future for phenomenographic research? On continuity and development in the phenomenography and variation theory research tradition. *Scandinavian Journal of Educational Research*, *62*(6), 949-958. https://doi.org/10.1080/00313831.2017.1324899

Bashir, G. M. M., & Hoque, A. S. M. L. (2016). An effective learning and teaching model for programming languages. *Journal of Computers in Education*, *3*(4), 413-437. https://doi.org/10.1007/s40692-016-0073-2

Bashan, B., & Holsblat, R. (2017). Reflective journals as a research tool: The case of student teachers' development of teamwork. *Cogent Education*, *4*(1), 1374234. https://doi.org/10.1080/2331186X.2017.1374234

Booth, S. (1997). On phenomenography, learning and teaching. *Higher Education Research and Development*, *16*(2), 135–158. https://doi.org/10.1080/0729436970160203

Breed, B., Mentz, E., Havenga, M., Govender, D., Govender, I., Dignum, F., & Dignum, V. (2013). Views of the use of self-directed metacognitive questioning during pair programming in economically deprived rural schools. *African Journal of Research in Mathematics, Science and Technology Education, 17*(3), 206-219. https://doi.org/10.1080/10288457.2013.839154

Carr, S. C. (2002). Assessing learning processes. *Intervention in School and Clinic*, *37*(3), 156, 7p, 2 charts.

Dasuki, S., & Quaye, A. (2016). Undergraduate students' failure in programming courses in institutions of higher education in developing countries: A Nigerian perspective. *The Electronic Journal of Information Systems in Developing Countries*, *76*(1), 1-18. https://doi.org/10.1002/j.1681-4835.2016.tb00559.x

Fasogbon, S. K., Jegede, P. O., Adetan, D. A., & Aderbigbe, A. A. (2016). Assessment of Java programming self-efficacy among engineering students in a typical Nigerian university. *African Journal of Sustainable Development*, *6*(2), 173-187.

Ferrari, A. (2012). *Digital competence in practice: An analysis of frameworks*. JRC Technical Reports. https://ifap.ru/library/book522.pdf

Govender, I., & Grayson, D. (2007). Pre-service and in-service teachers' experiences of learning to program in an object-oriented language. *Computers and Education, 51*(2), 874-885. https://doi.org/10.1016/j.compedu.2007.09.004

Govender, I., Govender, D. W., Havenga, M., Mentz, E., Breed, B., Dignum, F., & Dignum, V. (2014). Increasing self-efficacy in learning to program: Exploring the benefits of explicit instruction for problem solving. *TD The Journal for Transdisciplinary Research in Southern Africa, 10*(1), 187-200. https://doi.org/10.4102/td.v10i1.19

Hans, F., & Ellis, R.A. (2019). Using phenomenography to tackle key challenges in science education. *Frontiers in Psychology. 10*(1414), 1-10. https://doi.org/10.3389/fpsyg.2019.01414

Khan, Md. S. H, Abdou, B. O., Kettunen, J., & Gregory, S. (2019). A phenomenographic research study of students' conceptions of mobile learning: An example from higher education, *SAGE Open.* 2019, 1-17. https://doi.org/10.1177/2158244019861457

Kori, K., Pedaste, M., Leijen, Ä., & Tõnisson, E. (2016). The role of programming experience in ICT students' learning motivation and academic achievement. *International Journal of Information and Education Technology, 6*(5), 331–337. https://doi.org/10.7763/ijiet.2016.v6.709

Korkmaz, O., & Altun, H. (2014). Adapting computer programming self-efficacy scale and engineering students' self-efficacy perceptions. *Participatory Educational Research (PER), 1*(1), 20-31. https://doi.org/10.17275/per.14.02.1.1

Kwon, K. (2017). Novice programmer's misconception of programming reflected on problem-solving plans. *International Journal of Computer Science Education in Schools*, *1*(4), 14. https://doi.org/10.21585/ijcses.v1i4.19

Marton, F. (1986). Phenomenography: A research approach to investigating different understandings of reality, *Journal of Thought*, *213*(3), 28-49.

Marton, F., & Booth, S. (1997). *Learning and awareness*. Lawrence Erlbaum.

Niitsoo, M., Paales, M., Pedaste, M., Siiman, L., & Tõnisson, E. (2014). Predictors of informatics students' progress and graduation in university studies. In *International Technology, Education and Development Conference*. Valencia, Spain.

Özmen, B., & Altun, A. (2014). Undergraduate students' experiences in programming: Difficulties and obstacles. *Turkish Online Journal of Qualitative Inquiry*, *5*(3), 1-27. https://doi.org/10.17569/tojqi.20328

Prosser, M., & Millar, R. (1989). The "how" and "what" of learning physics. *European Journal of Psychology of Education*, *4*(4), 513–528.

Prosser, M., & Webb, C. (1994). Relating the process of undergraduate essay writing to the finished product. *Studies in Higher Education*, *19*(2), 125–138. https://doi.org/10.1080/03075079412331381987

Scherer, R., Siddiq, F., & Sánchez Viveros, B. (2020). A meta-analysis of teaching and learning computer programming: Effective instructional approaches and conditions. *Computers in Human Behavior*, *109*, 106349. https://doi.org/10.1016/j.chb.2020.106349

U.S. Department of Labor Bureau of Labor Statistics. (2018). *Occupational outlook handbook, 2017-2018 edition*. U.S. Department of Labor, Washington, D. C. https://www.bls.gov/ooh/computer-and-information-technology/home.htm

Vihavainen, A., Paksula, M., & Luukkainen, M. (2011, March). Extreme apprenticeship method in teaching programming for beginners. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (pp. 93-98). ACM. https://doi.org/10.1145/1953163.1953196

AUTHOR

Irene Govender is a Professor of Information Systems and Technology at the University of KwaZulu-Natal. She is an NRF rated researcher in the field of computing and technology for learning. Her field of research is a niche area of OOP programming, technology for learning, and ICT4D. She has 25 years in higher education, teaching security, networking, and OOP programming – 15 of which was specifically in teacher education for computer science. Prior to this period, she has been a teacher of Mathematics. She is widely published and has been the Academic Leader for Information Systems and Technology for the past six years. Served as reviewer for over 15 international journals. Served as Moderator of BSc (honors) degree program (at local accredited colleges) offered by London Metropolitan University in the UK, part of the review panel for Ghana BEd programme, as well as examiner for theses internationally. Has been chief examiner for matric senior certificate for IT for several years. She believes that teaching and learning is a partnership.