DOI: https://doi.org/10.4038/sjhs.v3i1.54

Received: 15 October 2022 *Accepted: 15 February 2023*

Modeling the ICT Teachers' Perspective Towards Teaching of Computer Programming at Secondary School Level Using Interactive Qualitative Analysis (IQA)

Perera, K. G. S. K.

susil.p@sliit.lk

School of Education, Faculty of Humanities and Sciences, SLIIT, Malabe, Sri Lanka.

Abstract

Computer programming is viewed and experienced as a subject cognitively challenging to students as well as teachers. The aim of this study was to determine the Information and Communication Technology (ICT) teachers' perspective towards teaching computer programming in order to comprehend how ICT teachers perceive teaching computer programming and factors that influence their work. Forty-seven ICT teachers participated in this qualitative study. The research method used was an analytical framework known as Interactive Qualitative Analysis to model the ICT teachers' perspective. The perspective was modelled in terms of affinities (factors) such as the programming curriculum, ICT resources, time, programming language, evaluation, students' performance, teachers' programming skills, teachers' pedagogical programming knowledge, student and professional development programs. Further, the interaction among these affinities was also modelled. Programming curriculum was found to be the most influential affinity which should be revised to decrease the cognitive load on students. Teacher's ICT knowledge was the most influenced(or influential???) affinity which can be improved by ICT resources, professional development programs and evaluation policy on students. Increase in computer: student ratio is also a contributing factor to students' achievement.

Keywords: Interactive Qualitative Analysis, Programming Curriculum, Teachers' Perspective of Computer Programming,

Introduction

Introduction of ICT as an optional subject to the Sri Lankan school system at (GCE (OL) took place in 2007 (National Institute of Education, 2008). It has been observed that the candidates' performance in answering questions in ICT paper in GCE(OL) is poor. Furthermore, the majority of ICT teachers who teach students of GCE(OL) are less comfortable with computer programming.

Instructional strategies in programming

Students also perceive programming as an uncomfortable experience and tend to develop negative attitudes (Korkmaz, 2014). Although students' IQ and mathematical skills seem to have a bearing on learning programming, how gender or nationality is significantly related to it is not supported in literature (Ala-Mutka, 2004). Poor instructional methods and overlooking of students' different learning styles lead to unsuccessful learning (Korkmaz, 2014). However, even teachers have found that teaching programming is a difficult task (Yang et al., 2015). Nevertheless, studies have not been carried out to find the reasons for such a difficulty. Traditional teaching methods based on lectures or demonstration of the use of language syntaxes are found to be frequently demotivating students. Therefore, pedagogy of programming must be changed to create more interesting game-like methods for program learning. With the support of multimedia building-blocks even younger students could learn programming concepts without much difficulty (Maloney et al., 2008), buthow to teach programming at secondary school level is yet to be solved. Teachers must be graduated in a subject where computer programming is seriously taught and in contrast, traditional in-service training to convert non-ICT teachers to ICT teachers would be unlikely to build a teacher capable of teaching programming.

Pedagogical Content Knowledge (PCK) of teachers plays an important role in teaching programming. PCK has been defined as the knowledge that allows teachers to transform their knowledge of the subject into something accessible for their students. This is basically how to teach a subject so that students understand the subject better, thereby producing a good learning outcome (Shulman, 1986;). PCK for programming addresses issues like reasons to teach programming, what concepts needed to be taught, how the instruction needs to be designed, what the most common difficulties students confront and common misconceptions they make (Saeli et al., 2011). The procedure that teachers generally adopt in teaching programming is teaching the vocabulary and the syntax of the language first and then guide students to develop programming strategies. The actual difficulty in learning programming is not set in learning the syntax or key words of a language but in the designing of the algorithm (Ala-Mutka, 2004).

Research conducted by Dag and Durdu (2020) found that knowledge and skill required by ICT teachers in teaching programming at secondary school level is limited. Failures in teaching computer programming is usually assigned to the methods adopted by teachers, particularly aspects like poor representation of the problems to be solved. Some teachers limit their teaching to explanation of theories behind programming, like syntax and use of keywords, without providing sufficient practical opportunities to solve problems and trying them out on computers. Sometimes it is not possible to do this due to the lack of computers to meet the standard student to computer ratio. Exposure to practical programming tasks is essential in understanding both syntax and semantics of programming. This hands-on experience comes first in pedagogy (Salleh et al., 2013).

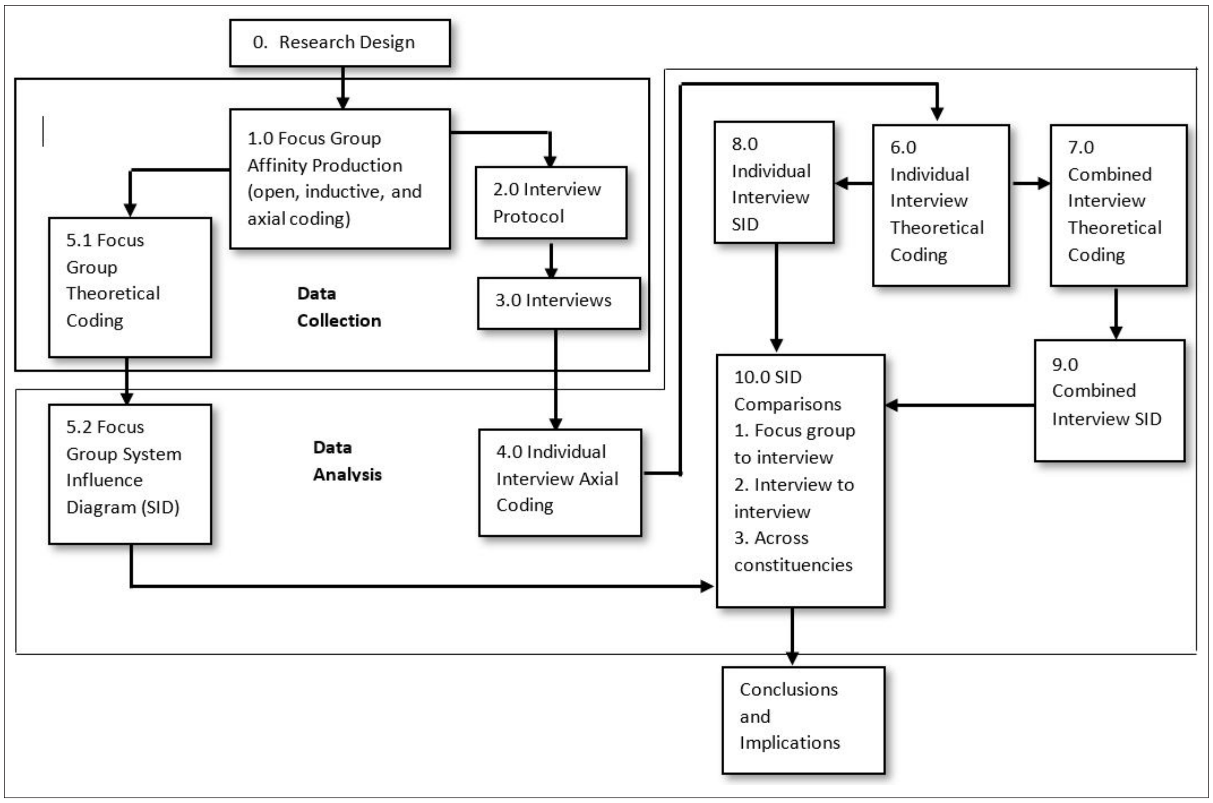
Problem-based teaching is also an effective method but must be coupled with laboratory experience. However, command style teaching is to be discouraged. For a better learning outcome one of the best approaches to follow is to identify the problem, determine inputs and outputs, design the algorithm, document the algorithm using flowcharts, conversion into pseudo code and finally coding using a programming language to test and debug (Sarpong et al., 2013; Garner, 2007). There is a scarcity of research regarding the pedagogy of computer programming such as how different teaching methods affect students despite the availability of several visual programming environments. However, different countries treat programming education at school level with varying weightages for programming concepts (Makris et al., 2013).

The best way to teach programming is believed to be that you begin at as lower grades as possible (European, 2015). In South Korea, starting from as early as Grade 7, programming includes more difficult topics as sorting, binary trees, graph traversals etcetera. However, Object Oriented Programming is not expected in this case. Teaching ICT, such as computer literacy and Computer Driving Licenses which are limited to the use of computers for general purposes instead of Computer Science, may not be helpful to students in Grades lower than 10 or so; to develop programming skills. In India, where computer education is not compulsory as in Sri Lanka, it is an elective subject from 9th Grade onwards (Jones et al., 2011). This was the situation in 2011 but it is expected to be changed in the years to follow. In Greece, at the inception of computer education at school level some of the teachers were direct recruits from those who graduated in computing. These teachers were not trained in pedagogy. The other teachers were those who were converted from other disciplines like Mathematics or Science subjects. This was a fast-track process and, while teaching they concentrated more on application of computers with a lesser emphasis on programming (Jones et al., 2011).

Interactive qualitative analysis (IQA)

IQA is a systems approach to qualitative research developed by Northcutt and McCoy of The University of Texas at Austin in 2004. IQA provides a framework to engage participants as a focus group and as individual interviewees to ground rich contextual data of the issue under study (Northcutt & McCoy 2004, p16). Follow-up interviews of participants are conducted to probe deeply into the constructs of the phenomenon surfaced via quantitative analysis of data (Bargate, 2014). One prominent advantage, among other things, of the IQA method is lowering of the researcher's direct involvement and minimizing of the subjectivity in interpretation of data. (Lasserre-Cortez, 2006). With IQA, constituents generate and interpret their own data while the researcher facilitates the process. The method involves generating data through two phases conducting a focus group session with participants and semi-structured individual interviews of participants. In this study IQA study answers at most two "generic" research questions: (i) what are the components of the phenomenon? and (ii) how do the components relate to each other in a perceptual system? (Northcutt & McCoy, 2004, p. 77). There are four stages of IQA research flow namely (i) research design, (ii) focus group brainstorming, (iii) interview, and (iv) report. In the first stage the research problem was enunciated, research questions were raised and participants were identified. Since of late, IQA is gaining popularity in qualitative research as an analytical frame to expound constituents' perspectives in phenomena under study (Bargate, 2014; Davis, 2019; Behling, Lenzi, & Rossetto,2022). The IQA research flow is illustrated by the figure1.

Figure 1.
Interactive qualitative analysis research flow (Northcutt & McCoy, 2004. p 45).


0. Research Design
1.0 Focus Group Affinity Production (open, inductive, and axial coding)
2.0 Interview Protocol
3.0 Interviews
5.1 Focus Group Theoretical Coding
Data Collection
5.2 Focus Group System Influence Diagram (SID)
Data Analysis
4.0 Individual Interview Axial Coding
8.0 Individual Interview SID
6.0 Individual Interview Theoretical Coding
7.0 Combined Interview Theoretical Coding
9.0 Combined Interview SID
10.0 SID Comparisons 1. Focus group to interview 2. Interview to interview 3. Across constituencies
Conclusions and Implications

Objectives of the study

In view of the above explanation, objectives of the study are to

- identify the contributing factors (affinities) towards the perceptions of the ICT teachers in teaching computer programming.
- develop a model to represent the perspective of ICT teachers towards teaching computer programming at GCE(OL).

Materials and Methods

Research design

The perspectives were studied employing IQA which is based on the systems approach. The design of the research with IQA consists of three significant steps: (a) statement of the problem, (b) definition of constituency groups and (c) the formulation of the research question (Northcutt & McCoy, 2004). According to the IQA framework, the perspectives are formed in terms of affinities towards the integration of ICT in the classroom and it is a qualitative approach.

Sample and instruments

The characteristics of the participants were determined according to the guidance in IQA. In IQA terminology, participants are called the focus group. It is defined as a "group of individuals who may certainly have varied opinions and experiences with the system under study but more critically share a common perspective" (Northcutt & McCoy, 2004, p 47). A sample of 47 ICT teachers were purposively selected from high, moderate, and less privileged government schools of Western, Southern and North-Western provinces. This sample size is sufficient for studies using IQA (Northcut & McCoy, 2004).

In the IQA framework, two instruments used were: (i) guidance for initial brainstorming on the expected 47 participants and (ii) the Interview Protocol (IP) which was the main instrument. The IP was utilized to interview participants and the content of the IP cannot be predetermined as it would be based on the affinities (categories of data) generated in the first stage of data collection process.

Data collection

In the first phase- silent brainstorming, the participants wrote down, on the cards provided, their perceptions (thoughts, feelings, reflections and experiences) of the issue being studied. Then these perceptions (data) were grouped and regrouped by the participants until further grouping was not logical. Each final group was labelled according to the nature of data and tilted as affinities. "Affinity is a set of textual references that have an underlying meaning or theme" (Northcutt & McCoy, 2004, p 81). Each affinity was provided with a description with the help of participants. This process is called open, inductive and axial coding in IQA (Ref. block number 1.0 of Figure 1). The information the participants provided was recorded in Individual Interview Axial Code Table (Table 1).

Table 1.
***Combined interview axial code table for affinity 1 (for example*).**

Teacher No.	Axial Quotation	Researcher's Note

The next step is referred to as theoretical coding in IQA in which pair-wise influences (relationships) of one affinity on another, if such influence exists, were determined. In this process, each participant was asked to identify influence pairs of affinities (affinity 1 influences affinity 2 etc.) and document them with reasons for each influence (Block number 5.1 of the Figure 1). Such documentation is illustrated in the Table 2.

Table 2.
Theoretical code affinity relationships.

Affinity Relationship	**Teacher No.**	**Theoretical Quotation**	**Researcher's Note**
1─►2	1	<reasons for the relationship>.	
	2	<reasons for the relationship>.	

The next step was to construct the Table 3. For this purpose, the number of teachers who had mentioned a given relationship between two affinities, in the same direction (eg. 1─►2), was counted and placed as frequency for that relationship. This table was the basis for analysis of data and this process is described in the block 5.2 of the Figure 1.

Table 3.
Affinities pairs in descending order of frequency with power analysis.

No.	Affinity Pair Relationship	Frequency (sorted in descending order)	Cumulative frequency	Cumulative Percent (Relation)	Cumulative Percent (Frequency)	Power

$$\text{Cumulative percent (relation)} = \frac{\text{number of cumulative relationships (cp)}}{\text{number of possible influence pairs (p)}} \times 100$$

Where $p={}^{N}P_{2}$ and N=number of affinities. cp =1,2,3... p

$$\text{Cumulative Percent (Frequency)} = \frac{\text{frequency for a relationship}}{\text{Cumulative frequency for all relationships}} \times 100$$

Power=Cumulative Percent(frequency)- Cumulative Percent(relation), which is an index of the degree of optimization of the system for a given relationship. Cut-off point of the Frequency Table (Table 3) is determined when the Power reaches the maximum as per MinMax criterion. IQA adopts the Pareto principle to statistically determine which of the inter-relationships should be included in the Interrelationship Diagram (IRD). The Pareto principle or 80/20 rule observes that 20% of the variables in a system will account for 80% of the total variation in outcomes in the system (Northcutt & McCoy, 2004).

Table 4.
***Interrelationship diagram*.**

Affinity	1	2	3	4	5	6	7	8	9	10	OUT	IN	Δ	Tentative assignment of the status to affinity	
1	▨	U	U	U	U	U	U	U	U	X	8	1		Type	Affinity name

Interrelationship Diagram (Table 4) was created as the first step to rationalize the system. This is called the tentative determination of drivers and outcomes**.** In this case, significant relationships of the Table 3 (1 to 43rd relationships) were tabulated in the Table 4. The "U" indicates that the left-hand side affinity (1) "influences" (2,3 etc.) and "X" indicates it is "influenced". "OUT" is the sum of "U"s and "IN" is the sum of "X"s for each affinity. Delta (Δ) is equal to OUT-IN. This categorization assists in the topology design of cluttered System Influence Diagram (SID).

The relationship pairs at and above where the Power reached maximum were considered to construct concept map like relationship (among affinities) diagram which is titled as Cluttered System Influence Diagram (Cluttered SID). This diagram is optimized to arrive at uncluttered SID by removal of redundant relationships among affinities. Redundant link is defined as "a link between two affinities in which, even if removed, a path from the driver to the outcome can be achieved through an intermediary affinity" (Northcutt & McCoy, 2004, p 178). Uncluttered SID is useful to simplify the system and optimise its explanatory power. The cluttered SID was used as a basis for preparing the script for individual interviews in order to get an insight into the issue under study (block 10 of the Figure 1).

Results and Discussion

The 47 participants produced 224 ideas. After inductive and axial coding, ten affinities were identified as (1) Programming Curriculum, (2) ICT Resources, (3) Time, (4) Programming Language, (5) Evaluation, (6) Performance, (7) Teacher's Programming Skill, (8) Teacher's Pedagogical Knowledge on Programming, (9) Student and (10) Professional Development Program. Table 5 represents a sample of axial quotations for the affinity programming curriculum.

Table 5.
Combined interview axial code table for affinity programming curriculum (sample).

Teacher No.	Axial Quotation	Researcher's Note
1	Subject matter should be sequenced to attract students' attention and help them build programming skills.	
2	Programming syllabus is too difficult for students to follow.	

In the next step, each participant was asked to identify influence pairs of affinities and document them with reasons for each influence. Table 6 illustrates a sample of Theoretical Code Affinity Relationships.

Table 6.
Theoretical code affinity relationships.

Affinity Relationship	**Teacher No.**	**Theoretical Quotation**	**Researcher's Note**
1─►2	1	Programming curriculum prescribes the type of operating system needed.	
	2	Programming practical cannot be done without the recommended programming language software.	

Frequencies (how many participants have mentioned a particular relationship between two affinities) were calculated and tabulated in the descending order of frequency. Then the Table 7 was constructed as per the directions in IQA.

Table 7 illustrates calculation of Power and cut-off point to consider the relevant relationships. (example 1─►4 means Programming Curriculum influences Programming Language (to be selected for the curriculum).

Table 7.
Affinities pairs in descending order of frequency with power analysis.

No.	Affinity pair Relationship	Frequency Sorted (Descending)	Cumulative Frequency	Cumulative Percent (Relation)	Cumulative Percent (Frequency)	Power
1	1─►2	47	47	1.111	1.836	0.7248
2	1─►3	47	94	2.222	3.672	1.4497
3	1─►4	47	141	3.333	5.508	2.1745
Rest is here						
42	7─►10	35	1815	46.667	70.898	24.2318
43	4─►1	29	1844	47.778	72.031	24.2535
44	6─►8	22	1866	48.889	72.891	24.0017
Rest is here						
90	10─►1	10	2560	100	100	0

The cut-off point taken from the entire data set of Table 7 was at 43rd relationship where the power has reached its peak. It can be observed that the first 43 relationships out of 90 i.e. number of permutations or ${}^{10}P_2$ (47% of the total) account for 72% of the total variation to be significant to construct the cluttered System Influence Diagram (SID). In fact, in this study

Power was maximum at 72% and it was used as the cut-off point as it is closer to 80% (Pareto statistics). Interrelationship Diagram was created as the first step to rationalize the system. In this case, significant relationships of the Table 7 (1 to 43rd relationships) were tabulated in the Table 8. This table was later sorted in descending order of delta.

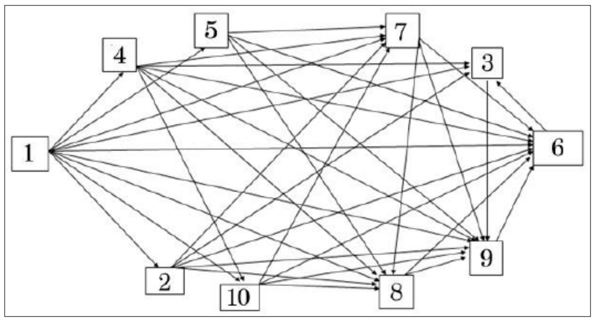
Table 8.
Interrelationship diagram.

Affinity	1	2	3	4	5	6	7	8	9	10	OUT	IN	Δ	Tentative Assignment of the status to affinity	
1		U	U	U	U	U	U	U	U	U	8	0	8	PD	Programming Curriculum
2	X		U			U	U	U	U		5	1	4	SD	ICT Resources
3	X	X		X		X		X	U		1	5	-4	SO	Time
4	X		U			U	U	U	U	U	6	1	5	SD	Programming Language
5	X					U	U	U	U		4	1	3	SD	Evalution
6	X	X	U	X	X		X	X	X	X	1	8	-7	SO	Performance
7	X	X		X	X	U		U	U	X	3	5	-2	SO	Teacher's Programming Knowledge
8	X	X	U	X	X	U	X		U	X	3	6	-3	SO	Teacher's Pedagogical Programming Knowledge
9	X	X		X	X	U	X	X		X	1	7	-6	SO	Student
10	X			X		X	X	X	X		4	2	2	SD	Training

(PD = Primary Driver; SD = Secondary Driver; SO = Secondary Outcome)

According to the Table 8, the topology of the cluttered SID (Figure 2) was decided and drawn: PDs at the extreme left, SDs in the middle and SOs at the extreme right. In other words, order of the affinities was from left to right depending on the value of the affinities (highest on the left and lower values from left to right).

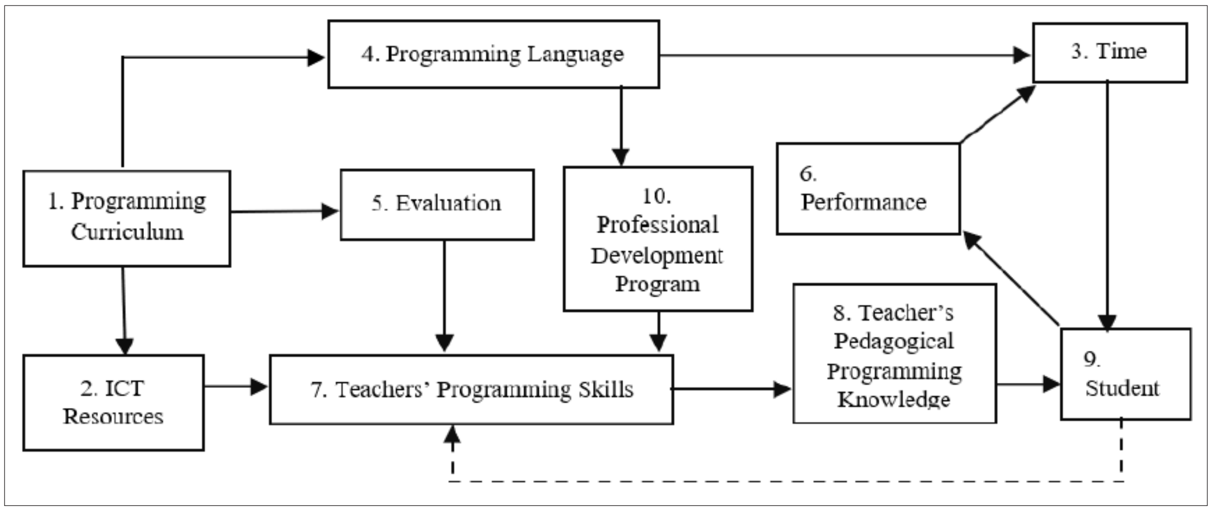
Figure 2.
Cluttered systems influence diagram.



This diagram was used as the interview guide to conduct individual interviews with participants on the affinities and their relationships in order to collect qualitative data on the participants' perspective on teaching programming. Out of 47 participants only 35 were available for interviewing. Participants were asked to describe each affinity and explain the relationship between affinity pairs as per their perspective of teaching of programming. Interview data was used for further clarification of the affinities and the relationship among them (Modules 2.0, 3.0, 4.0, 6.0, 7.0, 8.0 and 9.0 of the Figure 1).

The cluttered SID was refined by removing the redundant links to arrive at the **Uncluttered SID** (Figure 3). The uncluttered version of the SID is built to simplify the SID and bring more explanatory power to the diagram.

Figure 3.
Uncluttered systems influence diagram.



4. Programming Language
3. Time
6. Performance
1. Programming Curriculum
5. Evaluation
10. Professional Development Program
8. Teacher's Pedagogical Programming Knowledge
9. Student
2. ICT Resources
7. Teachers' Programming Skills

The Figure 3 represents the perspective of teachers towards Teaching of Computer Programming at Secondary School Level. The surfaced affinities and how one affinity influences another, either directly or indirectly, is indicated by arrows. The dotted line indicates a feedback loop i.e. Student can influence Teacher's Pedagogical Programming Knowledge through Teacher's Programming Skills

Descriptions of the 10 affinities

Following are the descriptions of the affinities from the participants' point of view resulted from combining the individual interviews held with the participants.

1. Programming curriculum

This is the subject matter on programming logic and the language recommended to be used to code the algorithms. In the current syllabus, more emphasis is on the language which is Visual Basic Version 6 (VB 6) (Microsoft, 1998) than the programming logic. As a result, less time is devoted for algorithms which are represented in flow charts and pseudo code. Teachers are in the opinion that current amount of subject matter is too much to cover within the time allocated since it demands more practical hours. Although students seem to be enjoying the visual programming environment, it is doubtful whether it adds value to them in building programming logic. Teachers complain that in the development of programming curriculum very little attention is paid to the other contributing factors (other affinities) to the success of teaching subject matter in the classroom.

2. ICT resources

Resources in this case are computers and programming software. Computer laboratories provided about 10 years ago lead toproblem of breakdown of computers without sufficient repair, thereby forcing sometimes four or five students to share one computer. This situation is not helpful when extensive hands-on experience is required for students to comprehend the subject well. Availability of a multimedia projector which is not a luxury enjoyed by every school, is also a good teaching aid in teaching programming.

3. Time

For a week, only three periods of 30 minutes each are allocated for the ICT subject at GCE(OL) and programming is taught in the grade 11 or the second year of G.C.E. (O/L.). The total subject matter for programming prescribed for Grade 11 is too heavy and time is not sufficient to cover the syllabus. Time is basically what is proposed by the syllabus, but the actual time is less than that due to various problems in the process. Time, which is lost due to holidays, delay in setting up of students at the computers, computer seizures and time lost in coming to the lab are some reasons for that. Time needed for teachers to be prepared for lessons is also regarded as Time in this case. Teachers complained that they lose valuable time as they are supposed to support administrative functions of the school where ICT applications are required.

4. Programming language

Currently the programming language is prescribed in the syllabus is VB 6 which provides a visual support to students when they learn programming. This is viewed as a supportive environment to learn programming. On the other hand, as this is an application developer's language with numerous facilities to expedite programming, students are likely to fail in converting a flow chart or pseudo code into computer program. Hence, it is required to use a suitable teaching language for this purpose. If it is a WYSIWYG type or programming language, which provides outcome of the program being executed immediately juxtaposed with the code typed in the editor of the programming tool, students would get a solid knowledge about programming.

5. Evaluation

Evaluation in this respect is the national examination conducted by the Examination Department of Sri Lanka at G.C.E. (O/L). There seems to be a mismatch between the subject matter and the questions on programming. Students are evaluated only on flow charts or pseudo codes but not on the coding language for which teachers make much effort to teach in the classroom. The only factor the evaluators consider is the programming curriculum but not any other contributing factor (affinity) emerged in this study. However, some teachers believe that fair questions are set to offset the difficulties faced by many students in a variety of school set-ups. Although about 1/3 of the syllabus is allocated to programming, only one essay type question appears in the question paper which is an unjustifiable proportion. On the other hand, programming questions, sometimes spanning across one and half pages of the question paper, could distract students in selection and discourage answering them.

6. Performance

Performance is referred to as how successful students are when they answer programming questions at G.C.E. (O/L) examination. Few marking examiners in the sample had observed that in most cases very few candidates attempt to answer programming questions and they either answer programming questions very successfully or very weakly. Inadequate knowledge on programming concepts may be the reason behind this disappointment.

7. Teacher's programming skills

Teacher's programming skills consist of knowledge of algorithms and coding a given algorithm in prescribed computer language in the syllabus. The participants admitted that this is a challenging part in the syllabus. Fewer attempts are made by teachers to develop algorithmic skills as the majority of the subject matter covers the use of programming language. It is the common agreement that an in-depth knowledge on programming is essential to teach the subject content.

8. Teacher's pedagogical knowledge on programming

This is basically about how to teach programming using suitable instructional strategies. No participant knew any such strategies. Demonstration followed by exercises, answers to which are discussed later is the general practice. This may be best suited for teaching how to use the programming language only. Participants agreed that problem, solution to it in a flow chart, writing pseudo code from the flow chart, coding it with the programming language and finally executing the program on the computer should be the right way to teach programming.

9. Student

Students who learn programming are perceived as the most significant affinity of this scenario. It was highlighted that students like to use the visual programming language packages to play around changing properties of the objects like forms, text boxes and buttons rather than implementing algorithms. Although knowledge of English would not be a big issue for students in learning how to use the programming language, it could be a drawback on students in the area of flow charting and pseudo coding. However, majority of students are found to be grappling with learning of programming.

10. Professional development programs

Professional development programs should be conducted either centrally by NIE, MOE or by in-service advisors at zonal education level in order to enhance the quality of teachers in the area of programming and its pedagogy. The impact of such programs is deferred but, participants did not deny the value-adding nature of such programs. Although subject matter is properly dealt with in these sessions, very little attention is paid to the pedagogical aspects of programming. Teachers preferred face-to-face type of professional development sessions on programming concepts, use of programming language for coding and instructional design on programming concepts and coding.

Results of the interviews guided by the cluttered SID

In order to make teachers more effective in teaching, the significant affinities are teachers' programming skills and their pedagogical programming knowledge. Pedagogical knowledge is only influenced by the programming knowledge which in turn is directly influenced by questions set in the ICT question paper of the GCE(OL) examination, ICT resources available to them in ICT laboratories and professional development programs for teachers. Teachers are compelled to update their programming knowledge having taken the feedback from students' display of programming knowledge which might have been acquired via alternative means like the Internet. Two directly influencing factors on students are time and teachers' pedagogical programming knowledge. Complexity of programming language prescribed by the curriculum demands more time to cover the programming content and on the other hand, poor performance of the students at GCE(OL) drives teachers to allocate more time for teaching of programming. In this scenario teachers have viewed that students are directly responsible for their performance and remaining affinities are only indirectly responsible for their success. The circular relationship of students─►performance ─►time─►students (Ref. Figure 3) when coupled with the findings in group and individual interviews with participants indicates that the poor performance of students demand increase in instructional time which is in turn helpful to students.

Teachers are not happy about ICT resources available for teaching programming. They commented "The number of computers is not enough for practical and as a result more than 2 students have to share a computer". They think time allocated for programming section is not sufficient as evident from "There are too many topics in the syllabus to cover and time allocation is not enough to teach algorithms and practical". They believe prescribed programming language is not suitable. They commented "Students tend to play with object in the IDE and it obliterates logical reasoning needed for programming". Teachers are in the opinion that students display poor performance in programming at GCE (OL) and it is believed that students are reluctant to learn programming. Teachers were neutral about the programming questions in GCE (OL) ICT question paper, teachers' pedagogical programming knowledge and professional development programs. The only aspect they were positive about is their own programming knowledge.

Teachers believe that programming curriculum influences all the other affinities. ICT resources required, allocation of time, type of programming language, programming questions of ICT question paper and students learning curve are determined by the programming curriculum. It also dictates teachers' pedagogical practice and nature of professional development programs. ICT resources facilitate teachers' pedagogical practice and students' performance. Selection of computer languages should not levy unnecessary overhead on both teachers' practice and students' performance. This is supported by the comment "Students should get a simple IDE where they can type codes and test them easily". How programming knowledge is evaluated could be an eye-opener for teachers to update and upgrade their programming and its pedagogical knowledge as evident from "In the exam what is tested is not coding but logic behind the solution to problems. Usually in the exam problem solving is not asked but fill-in the blank of a program is expected.". Teachers complain that poor performance of students at GCE (OL) examination push them to conduct after-school classes to gain more time for teaching. They have not underestimated the significance of professional development programs since such programs provide the opportunity for teachers gain both programming and pedagogical programming knowledge that eventually could ensure a successful instructional process in the classroom. However, they complain that "Enough in-service training is not provided on how to teach programming".

It is suggested that teachers should undergo comprehensive professional development programs that address programming knowledge and relevant pedagogical knowledge. One teacher complained that "Sometimes I feel embarrassed when certain students bring problems that I cannot write programs easily". The ICT curriculum must be reviewed to reconsider the programming language being used at present. Student-centered pedagogy must be encouraged, and students should be involved in solving more programming problems.

Conclusions and Recommendations

The goal of this study was to model the perspectives of the ICT teachers towards teaching of computer programming at GCE (OL). The affinities emerged were programming component of the ICT curriculum, ICT resources, time, programming language, evaluation at national level, performance of students of programming questions at GCE (OL), programming skills of the teachers, pedagogical programming knowledge of the teachers, students and professional development programs of ICT for teachers.

Most influencing aspect of the teachers' perspective is the programming curriculum which needs to be reduced in content. To make teachers more effective in teaching the significant affinities are teachers' programming knowledge and their pedagogical programming knowledge. Teachers are compelled to update their programming skills to meet students' demands for which they expect effective professional development programs. Two directly influencing factors on students are time and teachers' pedagogical programming knowledge. Complexity of programming language prescribed by the curriculum demands more time to cover the programming content and on the other hand, poor performance of the students at GCE(OL) drives teachers to allocate more time for teaching of programming. Selection of computer languages should not levy unnecessary overhead on both teachers' practice and students' performance. Lack of ICT resources in the school sector is a cause of poor coding skills of the students although coding skills are not evaluated at the GCE(OL) examination. However, how programming knowledge is evaluated could be an eye-opener for teachers to update and upgrade their programming and its pedagogical knowledge.

Based on the findings of the study, several recommendations are made. The programming curriculum should be reduced in content with a suitable replacement for the current coding language and the teachers should be instructed to spend more time on teaching algorithms than coding. Professional development programs should pay more attention to pedagogical programming knowledge for teachers. Authorities should ensure that number of computers in school ICT laboratories should be increased so that at least two students can share a computer for coding purposes.

References

Ala-Mutka, K. (2004). Problems in learning and teaching programming-A literature study for developing visualizations in the Codewitz-Minerva Project. *Codewitz Needs Analysis*, 1-13. Retrieved from: www.cs.tut.fi/~edge/literature_study.pdf .

Behling, G., L.enzi, F. C., & Rossetto. C. R. (2022). Upcoming Issues, New Methods: Using Interactive Qualitative Analysis (IQA) in Management Research. *Journal of Contemporary Administration,* 26(4), 1-18. https://doi.org/10.1590/1982-7849rac2022200417.en.

Chandrakumara, D. P. S. (2015). Regional Imbalances in the Distribution of Educational Resources in Sri Lanka. *International Journal of Applied Research,* 1(11), 13-21.

Dağ, F., & Durdu, L. (2020). Teacher Views on Programming Teaching. *Adiyaman Univesity Journal of Educational Sciences*, 10(2), 70-86. doi:http://dx.doi.org/10.17984/adyuebd. 629428.

European, S. (2015). *Computing our future.* Retrieved from http://www.eun.org/documents/411753/817341/Computing+our+future_final_2015.pdf/d3780a64-1081-4488-8549-6033200e3c03.

Garner, S. (2007). A program design tool to help novices learn programming. *Proceedings ascilite Singapore*, 321-324.

Jones, S. P., Stephenson, C., & Bell, T. (2011). Computing at school: International comparisons. Microsoft research UK, Version 5. Retrieved from http://www.computingatschool.org.uk/data/uploads/internationalcomparisons-v5.pdf.

Korkmaz, Ö. (2014). A validity and reliability study of the Attitude Scale of Computer Programming Learning (ASCOPL). *MEVLANA International Journal of Education.* 4(1), 30- 43, http://dx.doi.org/10.13054/mije.13.73.4.1.

Lasserre-Cortez, S.(2006). A Day in The Parc: *An Interactive Qualitative Analysis 0f school climate and teacher effectiveness through professional action research collaboratives.* [Unpublished doctoral dissertation]. The Department of Educational Leadership Research and Counseling. Graduate Faculty of the Louisiana State University and Agricultural and Mechanical College.

Maloney, J., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by Choice: Urban Youth Learning Programming with Scratch. Retrieved from http://web.media.mit.edu/~mres/papers/sigcse-08.pdf.

Makris, D., Euaggelopoulos, K., Chorianopoulos, K., & Giannakos, M. (2013). *Could you help me to* *change the variables? Comparing instruction to encouragement for teaching programming*. WiPSCE '13, Aarhus, Denmark. 11-13. doi:10.1145/2532748.2532761.

Microsoft. (1998). *Microsoft Visual Basic 6*. https://www.microsoft.com/en-US/Download/confirmation.aspx?id=10019.

National Institute of Education (2008). *Information and Communication Technology Syllabus: Grade11*. Maharagama, Sri Lanka.

Northcutt, N., & McCoy, D. (2004). Interactive Qualitative Analysis: A Systems Method for Qualitative Research. California: *Sage Publications* Inc, https://doi.org/10.4135/9781412984539.

Saeli, M., Perrenet, J., Wim, M. G., Jochems, W. M .G., & Zwaneveld, B. (2011). Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective. *Informatics in Education*,10(1), 73-88.

Salleh, S. M., Shukura, Z., & Judib, H. M. (2013). *Analysis of Research in Programming Teaching Tools: An Initial Review*. 13th International Educational Technology Conference. , Sakarya Universitesi, Turkey doi: 10.1016/j.sbspro.2013.10.317.

Sarpong, K. A., Arthur, J. K., & Amoako, P. Y. O. (2013). Causes of Failure of Students in Computer Programming Courses: The Teacher - Learner Perspective. *International Journal of Computer Applications,* 77(12), 27-32.

Shulman, L. S. (1986). Those who understand: knowledge growth in teaching. *Educational Researcher*, 15, 4-14.

Yang, T. C., Hwang, G. J., Yang, S. J. H. & Hwang, G. H. (2015). A Two-tier test-based approach to improving students' computer programming skills in a Web-based learning environment. *Educational Technology & Society*, 18(1), 198-210.