

# Application of Modified Perceived Learning Problem Inventory (PLPI) to Investigate Performance in Introductory Programming

Paul Golding, Opal Donaldson, and Vanesa Tennant

University of Technology Jamaica, pgolding@utech.edu.jm, odonaldson@utech.edu.jm, vtennant@utech.edu.jm

**Abstract** – Programming has been recognized by universities as a complex and difficult intellectual activity, with students struggling through their first programming subject and educators struggling to teach it. Several universities have embarked on using innovative practices to improve students' performance in introductory programming. To no avail many students still fail. This paper postulated that a holistic approach should be used to evaluate the factors contributing to the low pass rate in an introductory programming course. The authors developed a modified version of Perceived Learning Problem Inventory (PLPI) to evaluate student performance. The model suggested that the factors that affect student performance in programming squarely lies with the subject content and other factors play no role in determining performance. The findings are consistent with earlier research which indicated that problem solving skills, mathematical capabilities, logical reasoning and previous programming experience as contributing factors to failure.

*Index Terms* – Introductory Programming, PLPI, Abstraction

## INTRODUCTION

Programming is central to any Information Technology and Computer Science bachelor's degree in a University. In the first year of study, students are normally introduced to an Introductory Programming course which serves as the foundation for related courses. Programming has been recognized by universities as a complex and difficult intellectual activity, with students struggling through their first programming subject and educators struggling to teach it [1]. As noted by Hamilton, Harland, Muir et al, schools/departments make attempts to teach programming well and allocate disproportionate resources to ensure successful, long-term outcomes, in terms of building students' skill levels and confidence in programming [2]. However, the authors noted that despite such attempts and allocation of resources, programming continues to pose early-learning challenges for students with little or no programming background. The literature has over the years highlighted that introductory programming courses tend to

have a high attrition rate. With reference to [3], a study by [4] sought to validate this concern with a survey of universities and colleges worldwide. Although the findings could not confirm that failure rates were abnormally high in programming units, they did find that pass rates were on average 67%. It has been asserted by [5] that the problem arise as students tend to struggle with developing algorithms, figuring out how to apply problem solving techniques in their programs, and using common programming constructs. The suggestions to solve this long standing issue include peer tutoring [6]; use of alternative programming languages [7] [5]; Pair programming [8]; correlation between mathematics courses and programming [9] [10] and others.

This research takes a different approach by applying the modified Perceived Learning Problem Inventory (PLPI) originally proposed by [11] to investigate the performance of first-year programming course. The factors included Personal Behaviour Characteristic (sub-divided into personal problems with examinations, psychological problem of students, lack of commitment to study, student learning problem), lecturer/tutor behavior characteristics, and content of subject. The remainder of the paper is outlined as follows: Section 2 provides a review of the literature guiding the research; Section 3 discusses the methodology, and Section 4 and 5 reports the finding and discussion respectively and Section 6 provide the conclusion.

## LITERATURE REVIEW

The art of programming was highlighted by [12] as the knowledge of programming tools and languages, problem-solving skills, and effective strategies for programming design and implementation. Empirical evidence has been presented by several researchers identifying that learning to program is difficult for many students [13] [12] [14]. It was suggested that the dropout and failure rates in introductory programming courses at the university level is evidence of the difficulties associated with programming [13]. Ref [15] corroborates this point highlighting that most persons can not learn programming since between 30% and 60% of every computer science department's intake fails the first programming course. Ref [15] cautioned that experienced

teachers are weary and oblivious to most people’s inability to learn programming while new beginners are convinced that the wrong strategies have been used in the past; however they soon face the reality based on experience in the classroom. Ref [16] proposed that one of the main problems is that students are vulnerable to computer infatuation based on a mode of user rather than developer or problem solver. From another standpoint [13] purported that although several factors affecting student programming learning have been identified over the years there is a far way to go in terms of understanding why some students learn to program quickly and easily and others flounder (see table 1).

TABLE 1: RESEARCHER’S CLASSIFICATION ON THE SKILLS REQUIRED FOR PROGRAMMING

Author	Skills Required for Programming
[17]	Problem solving skills & Mathematical thinking capabilities
[13]	Self-efficacy, Previous Programming Experience & Mental Models
[15]	Mental Models (using pretest for programming aptitude)
[18]	Problem Solving Skills, Logical Reasoning, Coding and Debugging, and previous programming experience
[14]	Problem Solving Skills, Abstraction, Knowledge of syntax

*Difficulties faced by Students in Introductory Programming*

The difficulties faced by students in the learning programming has been cited as not a lack of motivation on anybody’s part; since students join these courses with the interest in understanding the concepts to be taught and teachers are keen to help them [15] [19]. In corroboration with sentiments expressed by [15], [16] pointed out a high level of frustration expressed by student in the words “I hate programming”. It was further articulated by [15] that for reasons unknown most students struggle on to the end of the course while those who can program are frustrated by the slow pace of teaching. Citing [20], [15] explained that student behavior towards programming can be classified in to three categories:

- Stoppers: In problematic situation the tendency is to abandon all hope of solving the problem on their own
- Movers: In problematic situation they keep trying, modifying code and using feedback on errors effectively
- Tinkers: Cannot track their program, make changes more or less randomly and like stoppers do not make much progress

Ref [14] explained that students face the difficulty of imagining and comprehending abstract terms such as variables and data types, which do not relate to real life. The

authors also posit that some students struggle with learning the correct syntax required working in a particular programming environment. Making reference to [21], [16] cited that the core problems faced in learning programming includes:

- Complexity: The number of programming details that students must master has grown faster than the corresponding number of high level concepts
- Instability: The languages, libraries and tools are changing more rapidly than in previous years.

On the other hand, [16] explained that results of their study suggest that student’s propensity to make errors during programming is largely associated with their academic levels. The authors point out that increase exposure to problem solving over time improves the ability to program. Considering all the problems highlighted in previous research, [12] stated that many students have difficulties in identifying their deficiencies in learning to program. The author pointed out that while students believe themselves that they have understood the issues teachers still find shortcomings in their programming coursework and examinations. It is also important to note that main source of difficulty do not appear to be syntax or understanding of concepts but rather basic programming planning. Ref [17] explained that students’ lack of mathematical thinking capabilities is a deterrent to understanding programming. On the contrary, [22] suggested that although students have low mathematical skills they can be successfully taught programming after they have been had sufficient exposure to structured problem solving.

*Strategies Employed in Addressing Problems faced by Students in learning Programming*

To combat the problems faced by students in their quest to understand programming, several strategies have been used by intuitions. Some of the intervention use to help students develop programming skills includes: changes in curriculum, pedagogy and assessment [14]. One of the approaches used to improve students’ performance is peer-tutoring [23] [24]. A novel strategy has been created by [23] using an automated peer-assisted assessment of programming skills. The researchers utilized an automated system to validate students’ ability to conduct peer assessments. The results suggested that peer-assessment can play an important role in the improvements of students programming skills. The findings presented by [23] supports the arguments by [24] highlighting that students who practice peer programming techniques are more likely to persevere through introductory programming and receive a grade of C or better. Using another approach to improving students’ performance, [22] explained that a successful strategy used at Liverpool Hope University is the use of in-house books which complements lectures given; since prior experience has shown that many textbooks are not appropriate for the undergraduate level since it is assumed

that the reader has knowledge of problem solving. On the other hand [14] suggested that an approach to improving students understanding in programming is to integrate tutorials with lab sessions, which is intended to improve students interaction with tutors and their peers. Whatever the approach used to improve students' understanding of programming the holistic intent by academics and researchers is to decrease the attrition rate for introductory programming course.

### *Perceived Learning Problems Inventory*

The concept of the Perceived Learning Problem Inventory (PLPI) was introduced to investigate the perceived causes of failure among secondary students (high school) [11]. Ref [25] highlighted in her study that the poor performance of students in high school cannot be the result of any single factor but a combination that affects them at the personal level. In their study, [11] hypothesized that if students are given the opportunity to articulate the most salient factors perceived by them as impeding their ability to effective learning, they will most likely project real personal feelings. On this premise, they proposed the Perception of Learning Problems Inventory (PLPI) which they argued was capable of assessing both personal and contextual factors, and can discriminate effectively between "failed" and "passed" students. Five (5) factors/constructs were examined in the PLPI. The first factor examined 'Personal Behavior Characteristics' which encapsulates (a) Personal Problems in Exams (b) Psychological Problems, (c) Lack of Commitment to study and (d) Learning Problems. The second factor looked at 'Teacher Behavior Character', which is the effect of the students' perception of teacher's classroom behavior. The third factor inspected the content of the subject on the performance of the student. The fourth and final factor examined items associated with family and home characteristics.

It should be noted that although the original questionnaire was constructed for high schools, we argue that it is applicable in examining the problems associated with learning programming courses. In addition, most of the students that enter our programme have just exited high school.

### **METHODOLOGY**

Ref [11] questionnaire originally had five (5) categories and seventy-three (73). This paper modified the questionnaire with the following changes:

- The content of the subject was categorized that was significantly revised. The original questionnaire had 5 general questions and we added 10 content specific questions related to programming based on previous literature (See Table 1).

- We removed the category 'family and home structure' and the questions seemed related to the original target sample, which was high school students
- Minor revisions were made to the wording of the questions, as the questions were originally asked in a basic way with the target being high schools. Care was taken however to not lose the essence of the questions.
- In total, we truncated the number of questions to sixty-seven (67) and respondents were instructed to indicate the importance of the skill on a five-point Likert scale ranging from 1 (Disagree) to 5 (Agree). The original question by Al-Methen and Wilkinson (1992) had a three-point Likert scale.

Paper based survey was used and convenience sampling was used to collect data from the students. Surveys instruments were distributed to first year students. Two hundred (200) survey instruments were distributed and one hundred and twenty-seven (127) were received. However, only one hundred and eighteen (118) were usable. The completed surveys were returned over a two week period.

The data was analyzed using SPSS and a confirmatory factor analysis of the proposed six categories was conducted. A confirmatory factor analysis of the proposed four dimensions was conducted. The Principal Component Analysis extraction method was applied, while the rotation method used was Varimax with Kaiser Normalization. For purposes of ease and clarity, factors/items with low loadings ( $<.3$ ) was suppressed. The results of the Kaiser-Meyer-Olin measure of sampling with a coefficient of .741 suggest that the sample is middling. The Bartlett's Test of Sphericity with a significance of .000 indicates that the data does not produce an identity matrix and are therefore approximately multivariate normal and acceptable for factor analysis.

An examination of the coefficient alpha values for the six dimensions indicates strong reliability values (all greater than the minimum value of .7) except for 3 factors (See Table 2). Lack of commitment had a value below average. Psychological problems had a weak value of 0.2751 and student learning problem had a value of .6774, hence all these were eliminated from the model. See Figure 1 for the model used in the analysis. Stepwise linear regression was also applied. The original questionnaire had grouped four sub categories under the grouping "personal behavior characteristics". However in the proposed model these four categories were disaggregated.

TABLE 2: CRONBACH ALPHA VALUES

Component/Factor	Cronbach's Alpha
Personal Problems with exam	0.7675
Psychological Problem	0.2751
Lack of Commitment to study	0.6651
Student Learning Problem	.6774
Lecturer/Tutor Behavior	.9372
Content of Subject	.9215

TABLE 3: AGE DISTRIBUTION GROUP

	Frequency	Percent
Valid		
16 to 20	43	36.4
21 to 25	34	28.8
26 to 30	24	20.3
31 to 35	11	9.3
36 to 40	4	3.4
41 & older	2	1.7
Total	118	100.0

A stepwise linear regression was used to analyze the data to determine the effect of each predictor variable on students' performance. The results showed that the construct subject content was the sole predictor of students' performance in Programming1 (see table 4 &5). The correlation (R) .339 depicted a relatively good incidence of subject content being a predictor variable for students performance with a p-value of .000 which is considered significant.

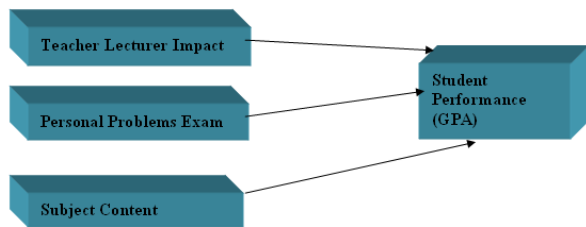


FIGURE 1  
RESEARCH MODEL

TABLE 4: MODEL SUMMARY

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.339(a)	.115	.107	2.54117

a Predictors: (Constant), SUBJECT\_CONTENT

TABLE 5: ANOVA SUMMARY

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	95.601	1	95.601	14.805	.000(a)
	Residual	736.158	114	6.458		
	Total	831.759	115			

a Predictors: (Constant), SUBJECT\_CONTENT

b Dependent Variable: Grade for programming 1

**FINDINGS**

*Demographics*

The gender distribution of the sample was 81 or 68.6% males to 37 or 31.4% females. This gender representation is considered representative of a typical computer science degree program cohort. This is supported by [26] who found that females tend to be underrepresented in university computer science courses. Majority 43 or 36.4% of the sample was within the age group 16 to 20. The least number of persons 6 or 5.1% were represented in the age group 36 and older (see table 3). The mode of study was equally distributed between part-time and fulltime students. Part-time students represented 60 or 50.8% of the sample while fulltime students accounted for 58 or 49.2%.

Further analysis of the construct subject content as it relates to gender differences indicated that a difference lies in the perception between genders on the level of mathematical knowledge possessed for programming1 (see table 6). Females had a higher mean score of 2.38 with a p-value of .039 while males had a lower mean score of 1.75. This suggests that males disagreed more than females that they had less mathematical knowledge than is required for the subject than females. In addition females also had a higher mean score of 2.03 in indicating their dislike for the subject with a p-value of .02.

TABLE 6: GENDER DIFFERENCES

	Gender	Mean	Std. Deviation	Std. Error Mean
Didn't like subject so i failed	Female	2.0270	1.44312	.23725
	Male	1.7778	1.16190	.12910
Not enough mathematical knowledge and aptitude	Female	2.3784	1.45967	.23997
	Male	1.7531	1.09008	.12112

The results showed that most students who received lower scores in programming 1 (grades C: U) had a lower mean score of 2.69 than students with higher scores (grade: C+: A) This indicated that persons with low scores disagreed more with the concept that too much focus was placed on programming during the course delivery. In addition, students receiving lower scores also disagreed more with the stance that they did not like programming and that's why they failed (see table 7).

TABLE 7: DIFFERENCES IN PERFORMANCE

Performance	Mean	Sig
Too much focus on Programming	low_performance	2.69
	high_performance	3.125
Didn't like subject so i failed	low_performance	1.589
	high_performance	2.375

**DISCUSSION**

This paper postulated that a holistic approach should be used to evaluate the factors contributing to the low pass rate in an introductory programming course. The authors developed a modified version of Perceived Learning Problem Inventory (PLPI) to evaluate student performance. The constructs included personal problems with examination (8 variables), psychological problems of students (3 variables), lack of commitment to study (10 variables), student learning problems (5 variables), teacher behavior characteristics (20 variables) and subject content (19 variables). After a confirmatory factor analysis, psychological problem, lack of commitment to study and student learning problems were excluded due to low Cronbach Alpha score of .2750. The modified model was evaluated using stepwise multiple regressions and only subject content was significant with a p-value of .000 and a correlation of 33.9.

The model suggested that factor that affect student performance in programming squarely lies with the subject

content and other factors play no role in performance. This finding is consistent with earlier research of [17], and [14] problem solving skills and mathematical capabilities, [18] logical reasoning and previous programming experience.

Further analysis was done to determine the difference in responses between genders. There was a significant difference between males and females on two questions or variables. Regarding the questions “I don’t like the subject because I keep falling” and “I don’t have enough mathematical knowledge and aptitude to apply to the subject” both males and females disagree; males having a higher level of disagreement with p-values of .039 and .02. Several researchers have highlighted that mathematics should be a prerequisite for programming [17], and [22]. It should be noted that [22] has suggested that students with low mathematical skills can still be successfully taught programming. Our study put forward that students do not attribute their performance to their knowledge of mathematics. However [12] argues that students have difficulties in identifying their deficiencies in learning to program.

Additional analysis was done to determine how performance impacted responses in the variables in content of subject. Two categories of performance were defined: Low performance was defined as grades C and below and high was above C. High performers were indifferent on the issues of “too much focus is placed on programming” (mean score 3.125) and a p-value of .012. However, low performers on this same question had a mean of 2.69. Both groups disagreed with the question “I don’t like the subject because I keep falling” high performers 2.375 and low performers 1.589 with a p-value of .000.

**CONCLUSION**

This paper examined the interaction effect of the different issues related to introductory programming subject content and the role of lecture/ tutor, Personal Problems with Exam and lack of commitment to studies. The results confirm existing empirical evidence presented by several researchers as it relates to the difficulties associated with programming. The result of the analysis is inconclusive and suggests that future work should be focused on examining the revised PLPI instrument with future cohorts to assess the validity of the instrument in assessing students’ performance in introductory programming course.

## REFERENCES

- [1] Lahtinen E, Ala-Mutka K, Järvinen, H “A Study of the difficulties of Novice Programmers”. 10th annual SIGCSE conference on Innovation and technology in computer science education ITiCSE '05., 2005
- [2] Daryl D., Hamilton, M., Harland, J., Muir, P., Thevathayan, C., Walker, C “Transforming learning of programming: a mentoring project”, *Proceedings of the tenth conference on Australasian computing education*, p.75-84, January 01-01, 2008
- [3] Butler, M., Morgan, M “The learning challenges faced by novice programming students studying high level / low feedback concepts”, ICT: Providing Choices for Learners and Learning, ASCILITE Singapore 2007
- [4] Bennedsen, J. & Caspersen, M. E. “Failure Rates in Introductory Programming,” *ACM SIGCSE Bulletin*, Volume 39 Issue 2 (pp. 32-36). 2007
- [5] Cooper, S., Dann, W., Pausch, R “ALICE: A 3-D tool for Introductory Programming Concepts”. Proceedings of the fifth annual CCSC northeastern conference on The journal of computing in small colleges Ramapo College of New Jersey, Mahwah, New Jersey, pp 107 – 116, 2000
- [6] Golding, P., Facey-Shaw, L. and V. Tennant, “Effects of Peer Tutoring, Attitude and Personality on Academic Performance of First year Introductory Programming Students”, Proceedings of the 36<sup>th</sup> ASEE/IEEE Frontiers in Education Conference, pp M2E7-12, 2006
- [7] Powers, K., Gross, P., Cooper, S. McNally, M., Goldman, K., Proulx, V., Carlisle, M. “Tools for teaching introductory programming: what works?”, Proceedings of the 37th SIGCSE technical symposium on Computer science education, 2006
- [8] McDowell C, Werner L., Bullock, H., Fernald, J "Pair programming improves student retention, confidence, and program quality " *Communications of the ACM* 49(8)., 2006
- [9] Bergin. S., Reiley, R. “Programming: factors that influence success.” *ACM SIGCSE Bulletin*, Volume 37, Issue, 2005
- [10] Golding, P., Donaldson, O. “Predicting Academic Performance”. Proceedings of the 36<sup>th</sup> ASEE/IEEE Frontiers in Education Conference, 2006
- [11] Al-Methen, A. E., & Wilkinson, W. J “Perceived causes of failure among secondary school students”. *Research in Education*, 48, 26-41, 1992
- [12] Ala-Mutka, K, “Problems in Learning and Teaching Programming- A literature Study for Developing Visualizations in the Codewitz-Minerva Project” *Codewitz Needs Analysis* 2004
- [13] Wiedenbeck, S, LaBelle, D, & Kain, V, “Factors Affecting Course Outcomes in Introductory Programming”, *16<sup>th</sup> Workshop of the Psychology Interest Group*. Carlow, Ireland, 2004
- [14] Miliszewska, I, & Tan, G, “Befriending Computer Programming: A Proposed Approach to Teaching Introductory Programming” *Issues in Informing Science and Information Technology* Volume 4, 2007
- [15] Dehnadi, S, & Bornat, R, “The Camel has Two Humps (working title)” *Middlesex University, UK* 2006
- [16] Naidoo, R, & Ranjeeth, S, “ Errors made by Students in a Computer Programming Course” *Proceedings of the Computer Science and IT Education Conference* 2007
- [17] Barros, Esteve, Dias, Pias and Soeiro “Using Lab Exams to ensure Programming Practice in an Introductory Programming Course” *Proceedings of the 8th annual conference on Innovation and technology in computer science education* 2003
- [18] Klassen, M, “Visual Approach for Teaching Programming Concepts” *9<sup>th</sup> International Conference on Engineering Education* 2006
- [19] Jones, M, “ The Redesign of an Introductory Programming Unit” *Italics* Volume 6 Issue 4, ISSN:1473-7507
- [20] Perkins, D., Hannock, C., Hobbs, R., Martin, F., Simmons, R “Conditions of learning in novice programmers.” In E. Soloway Spohrer and J.C editors, *Studying the Novice Programmer*, pp. 261-279. Lawrence Erlbaum Associates, Hillsdale, NJ 1989
- [21] Roberts. E “Resources to support the use of Java in introductory course. Proceedings of the 35<sup>th</sup> SIGCSE Technical Symposium on Computer Science Education, pp. 233-234. ACM Press., 2004
- [22] Whitefield, A, K, Blakeway, S, Herterich, G, E & Beaumont, C, “Programming, Disciplines and Methods Adopted at Liverpool Hope University” *Italics* Volume 6 Issue 4 2007
- [23] Lewis, S, & Davis, P, “The Automated Peer- Assisted Assessment of Programming Skills” *Unpublished School of Computing of Glamorgan* 2003
- [24] Williams, L, Yang, K, Ferzli, M & Miller, C, “In Support of Pair Programming in the Introductory Computer Science Course” *Unpublished North Carolina State University* 2003
- [25] Repole, D. “ Relationships between Anxiety, Coping, Perceived Learning Problems and Academic Performance in English Language and Mathematics among selected Grade 11 Jamaica students “ (Doctoral Dissertation, University of West Indies, 2008), 2008
- [26] Jones and Burnett “ Spatial Ability and Learning to Program” *Human Technology An Interdisciplinary Journal of Humans in ICT Environments* Volume 4 2008

## AUTHOR INFORMATION

**Paul Golding** is a Senior Lecturer in the School of Computing & Information Technology at the University of Technology, Jamaica. He is a member of the following organizations: IEEE, ACM, and AIS. His email address is [pgolding@utech.edu.jm](mailto:pgolding@utech.edu.jm)

**Opal Donaldson** is a Lecturer in the School of Computing & Information Technology at the University of Technology, Jamaica. She is a member of the AIS and can be contacted at [odonaldson@utech.edu.jm](mailto:odonaldson@utech.edu.jm)

**Vanesa Tennant** is a Lecturer in the School of Computing & Information Technology at the University of Technology, Jamaica. She is a member of the AIS and can be contacted at [vtenant@utech.edu.jm](mailto:vtenant@utech.edu.jm)