Learning Difficulties in Programming Courses: Undergraduates' Perspective and Perception

Phit-Huan Tan
Faculty of Information Science and Technology,
Multimedia University
phtan @ mmu.edu.my

Choo-Yee Ting
Faculty of Information Technology,
Multimedia University
cyting @ mmu.edu.my

Siew-Woei Ling
Faculty of Creative Multimedia,
Multimedia University,
swling @ mmu.edu.my

***Abstract* —** *Researchers have been searching for alternatives in teaching programming subjects. A reason to this is due to the fact that the compulsory subject in the field of Information Technology has been a challenge and they are tough subjects to learn. On top of that, lacking the understanding in concepts has reduced undergraduates' interests to pursue further exploration and self-experimentation. In this research work, a study was conducted to investigate the factors that lead to undergraduates' learning difficulty in programming courses and also their perception on which teaching methodology could be implemented to create richer and interesting learning process. The study involved 182 undergraduates from Multimedia University, Malaysia, who have taken the fundamental programming subject named Computer Programming I. The findings affirmed that undergraduates prefer to learn programming by referring to examples and using drill-practice method whereas learning via lecturing would only decrease their interest level. The challenge has provided an evidence to call for a better solution, game-based learning as an alternative to teach and learn computer programming subjects. Therefore, the authors proposed a game-based learning framework which consists of components that leverage the pedagogical aspects in designing game-based learning environment for programming subjects.*

Keywords - Programming; programming difficulties; learning; game-based learning; interest; motivation; education game

I. INTRODUCTION

The high dropout rate and failure rate in programming subjects have drawn serious attention of researchers to investigate the causes and solutions. Moser (1997) mentioned that programming is a multi-layers skill which is boring, intimidating and unrelated to day-to-day experience where students only learned in single context [1].

Most of the undergraduates do not have any programming experience before. Beginners start to learn programming in single context before learning structure and style. This might lead to a negative programming habit which may affect the flexibility of learning another programming language in different context. Besides, their wrong impression on learning these programming subjects leads to poor in mastering the skills. They think that programming is difficult to learn and use. Therefore, undergraduates might refuse to learn programming skills unconsciously.

II. JUSTIFICATION OF STUDY

It is important to highlight the causes that lead undergraduates to perform poorly in learning programming. Solution and alternative learning method could be implemented in order to assist them while learning programming. Undergraduates' perception on the learning problems and suggestions to improve teaching method could be helpful in identifying the alternative approach.

Many papers were reported in literatures about the problems and solutions of teaching and learning programming subjects. The findings of study are focus in foreign country such as Australia, Finland, United States and more in United Kingdom.

A study was conducted in Malaysia to investigate on the undergraduates' perception of learning problems and their perception as what may come out as the best method to learn programming language. This paper presents the challenges undergraduates faced when learning programming language and their perceptions on how learning should naturally take place.

III. METHODOLOGY

A. Questionnaire Design

One hundred and eighty two undergraduates taking an introductory subject to a programming course titled Computer Programming I in the Faculty of Information Science and Technology at Multimedia University were invited to participate in a web-based questionnaire. The questionnaire was modified from various existing instruments by Reek (2003), Lahtinen, Ala-Mutka & Hannu-Matt (2005), Milne & Rowe (2002) and Knezek &

978-0-7695-3892-1/09 $26.00 © 2009 IEEE
DOI 10.1109/ICCTD.2009.188

Christensen (1997), based on its relevancy to this study as shown in Table I.

The questionnaire consists of five parts that covers the demographic profiles, general computing background, computing programming experience, difficulties while learning programming and their experience in learning via game-approach. The results and findings from Part I to IV will be reported in this paper.

TABLE I. REFERENCE OF ITEMS IN QUESTIONNAIRE

Items	Description	Reference
Part I	Demographic profiles	-
Part II	General computing background	[2]
Part III	Computing programming experience	[2];[3];[4]
Part IV	Difficulties while learning programming	[3]
Part V	Experiencing with CeeBot-4	[5]

Items in Part II are adopted from Reek (2003). The General Computing Background and several items from Computer Programming Experience are extracted from Reek (2003) are adopted into the questionnaire of this study. Other items such as educational experiences, job experiences, program affiliation and learning style are omitted. Items in part III are referring to three different questionnaires from Reek (2003), Lahtinen, Ala-Mutka & Hannu-Matt (2005), and Milne & Rowe (2002). These questionnaires share the similar items with different combination of choices. Thus, they were combined and modified as items presented in Part III. All items from Lahtinen, Ala-Mutka & Hannu-Matt (2005) presented in their paper are adopted as items in Part III and IV. Items taken from Milne & Rowe (2002) are modified from 7-point Likert Scale to 5-point Likert Scale and combine with others to form items in Part III. The design of Part V is tailored from The Teachers' Attitudes Toward Computers Questionnaire version 5.1, the questionnaire from Knezek & Christensen (1997).

IV. RESULTS

This section reports on the data collected from Part I to IV of the questionnaire. The first part gathers the undergraduates' demographic profiles. There were 69.8% Malaysian and 30.2% International students. The female participants were 31.9% of total participants while 68.1% of them were male undergraduates.

The second part investigates the general computing background of undergraduates that help to determine their computer literacy. The first item explores the undergraduates' level of comfort of undergraduates in using computers. There were 18.7% of undergraduates claimed that they are uncomfortable when using computers while 81.3% of them were comfortable when using computers. The statistic shows that most undergraduates were getting fine when using computers. They were able to use computers for many activities including learning through reading and practicing.

The items shown in Table 2-4 are designed with 4-point Likert Scale (1, Never use; 2, Weak; 3, Moderate; 4, Strong).

TABLE II. LEVEL OF EXPERIENCE ON DIFFERENT COMPUTER PLATFORMS

	Mean	Std. Deviation
Windows	3.27	0.60
Linux/Unix	1.71	0.83
Macintosh	1.46	0.73

Table 2 illustrates that the undergraduates' level of experience on different computer platforms. Undergraduates had chosen their level of experience for the three platforms, namely, Microsoft Windows, Macintosh and Linux/Unix. As shown in the table, Microsoft Windows was the most popular operating system applied by undergraduates compare to others with mean = 3.27 and standard deviation = 0.60. Undergraduates rated their experience using Linux/Unix with mean = 1.71 and standard deviation = 0.83. There were 66.48% of the undergraduates replied that they never use Macintosh. The mean value for level of experience in using Macintosh is the lowest with mean = 1.46 and standard deviation = 0.73.

The third part of the questionnaire captures data about undergraduates' computing programming experience such as knowledge about programming language and topics.

TABLE III. LEVEL OF EXPERIENCE FOR DIFFERENT PROGRAMMING LANGUAGE

	Mean	Std. Deviation
C++	2.58	0.65
Visual Basic	2.27	0.89
C Language	1.76	0.92
Java	1.69	0.87
Javascript	1.41	0.69
C#	1.37	0.66
PHP	1.31	0.64
ASP	1.24	0.54
Pascal	1.14	0.40
Pearl	1.09	0.34
Fortran	1.08	0.32
Python	1.08	0.31

Table III represents the undergraduates' level of experience for different programming language. Undergraduates are low in confidence of their programming

skills. The average mean values shown in table were less than 3.00. The programming language C++ and Visual Basic were the programming language that used in teaching for several programming subjects in the faculty. Thus, undergraduates were more familiar with these language compare to others with mean = 2.58, standard deviation = 0.65 and mean = 2.27, standard deviation = 0.89 respectively.

TABLE IV. LEVEL OF UNDERSTANDING ON DIFFERENT PROGRAMMING TOPICS

	Mean	Std. Deviation
Variables (lifetime, scope)	2.78	0.68
Logic structures: decision (Selection structures, If, Switch)	2.76	0.67
Logic structures: iteration (Loop structures)	2.69	0.65
Input/output handling	2.55	0.75
Error handling	2.34	0.72
Arrays	2.30	0.62
Parameters	2.25	0.75
Structured data types	2.19	0.76
Using language libraries	2.12	0.70
Recursion	2.09	0.70
Abstract data types	2.05	0.73
Pointers/references	2.00	0.67
Encapsulation	1.85	0.78
Inheritance	1.83	0.80
Polymorphism	1.77	0.80

Table IV summarizes undergraduates' level of understanding on different programming topics. The sequence of topics in the table is listed from the highest mean value to the lowest mean value. Those topics with mean values above 2.00 are included in the syllabus of Computer Programming I while encapsulation, inheritance and polymorphism are topics covered in Computer Programming II.

In the fourth part, undergraduates need to response on the difficulties while learning programming, the effective situations and materials to facilitate in learning programming, and the factors that leads to poor in programming. 5-points Likert Scale (1, Strongly Disagree; 2, Disagree; 3, Neutral; 4, Agree; 5, Strongly Agree) is used on the items shown in Table 5-9.

TABLE V. DIFFICULTIES WHILE LEARNING PROGRAMMING

	Mean	Std. Deviation
Designing a program to solve certain task	3.52	0.99
Dividing functionality into procedures	3.51	0.88
Learning the programming language syntax	3.37	0.96
Finding bugs from my own program	3.34	0.92
Understanding basic concept of programming structures	3.18	1.01
Using program development environment	3.15	0.85
Gaining access to computers/networks	3.09	1.01

The difficulties faced by undergraduates while learning programming is shown in the Table V. This item evaluates the undergraduates' level of agreement. Undergraduates agreed that they were having problems while designing a program to solve certain task and hard to divide the functionality into procedures. Learning programming syntax and finding bugs in program could be the problems in learning programming. They have least problem on understanding basic concept of programming structures and using the development environment. Gaining access to computers/networks has the lowest mean value compare to others. This shows that this fact is not the main reason in making learning programming to be difficult for them.

TABLE VI. SITUATIONS THAT WOULD HELP TO LEARN PROGRAMMING MORE EFFECTIVELY

	Mean	Std. Deviation
In practical or lab sessions	3.98	0.91
Consultation or discussion with lecturers, tutors, seniors or friends	3.94	0.88
In small group exercise sessions	3.68	0.96
While working alone on programming coursework	3.41	1.17
In lectures	3.00	1.06

Most of undergraduates responded that they can learn programming more effectively in practical or laboratory sessions as shown in Table VI. Consultation or discussion with others might be helpful compared to small group

exercise sessions. Practicing could be more effective in learning programming as undergraduates rated that working alone on programming coursework is better than lectures.

TABLE VII. MATERIALS THAT WOULD HELP TO LEARN PROGRAMMING

	Mean	Std. Deviation
Example programs	4.35	0.70
Exercise questions and answers	4.00	0.88
Interactive visualizations	3.65	0.94
Programming course book	3.61	0.93
Still pictures of programming structures	3.59	1.01
Lecture notes/copies of transparencies	3.51	0.96
Forums	3.40	1.00

Examples programs were rated as the most useful materials to assist in learning programming. Most of the undergraduates agreed that exercise with questions and solutions may help them to learn programming. Interactive visualizations which provide more interaction is useful than programming course book. Undergraduates also gain knowledge via still pictures of programming structures, lecture notes/copies of transparencies and forums.

TABLE VIII. LEVEL OF AGREEMENT OF USING INTERACTIVE ENVIRONMENT IN LEARNING PROGRAMMING

	Mean	Std. Deviation
Interactive environment such as games will help in learning programming	3.59	0.95

There were 52.75% of the undergraduates gave positive respond in using interactive environment such as game in learning programming. They might think that practicing in game could be more helpful to learn programming.

Undergraduates stated that fewer examples in practical use may lead them to perform poorly while learning programming. The syllabi focuses and coverage were also rated as factors that make them felt lack of interest to learn programming subjects. The rest of the factors listed in the table were considered equally discouraging by undergraduates.

TABLE IX. FACTORS THAT LEAD TO PERFORM POORLY IN PROGRAMMING SUBJECTS

	Mean	Std. Deviation
Less examples in practical use are shown	3.68	0.96
Syllabi focuses too much on theory	3.62	0.90
Syllabi coverage per semester is too wide	3.60	0.95
Students lack of interest to learn	3.59	1.04
Learning environment that is not conducive	3.36	0.83
Presentation of instructors and their attention on students	3.36	0.87
Teaching methodology is less effective	3.35	0.87
Computers provided in labs are not functioning well	3.30	1.05

V. CONCLUSION AND LIMITATION

The findings revealed in this paper are aligned with major findings by other researchers. A study by Milne and Rowe (2002) revealed that learning programming has been difficult because the undergraduates' lack of understanding of how the program is executed. As such, a program visualization tool is suggested to be the solution to aid and enhance the programmer's understanding of what is happening in memory as their program executes [4]. Other studies such as Jenkins (1998) and Lahtinen et al. (2005) believe that one of the effective solutions to help in teaching and learning programming subjects is via practicing.

This study has several limitations. The findings are influenced by some characteristics. The participating undergraduates in this study were from same faculty in the university. Most of them were beginners in learning programming. Hence the results might be different if advance programmers were to take part in the survey.

The findings reported in this paper is crucial for the authors' further work, in which game can be another convincing teaching method that has a very conducive learning environment thus allows practical hands-on, provides various examples, provides feedback and hints, interaction and so forth in teaching and learning programming subjects [1, 4, 6-9]. Along with this study, authors had conducted an empirical study by utilizing an educational game, CeeBot-4, to determine the contributing game design components [10, 11] to promote the interest, motivation and fun among undergraduates to learn programming.

REFERENCES

[1] Moser, R. A fantasy adventure game as a learning environment: Why learning to program is so difficult and what can be done about it. in Proceedings of the 2nd conference on Integrating technology into computer science education. Sweden, 1997.

[2] Reek, M.M., C~ Computing Student Questionnaire, Rochester Institute of Technology and Department of Computer Science, 2003.

[3] Lahtinen, E., K. Ala-Mutka, and J. Hannu-Matt, A study of the difficulties of novice programmers, in Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education, ACM Press: Caparica, Portugal, 2005, p. 14-18.

[4] Milne, I. and G. Rowe, Difficulties in Learning and Teaching Programming—Views of Students and Tutors. Education and Information Technologies, 2002, **7**(1): p. 55-66.

[5] Knezek, G. and R. Christensen, The Teachers' Attitudes Toward Computers Questionnaire version 5.1., Denton, TX: University of North Texas and the Texas Center for Educational Technology, 1997.

[6] Natvig, L. and S. Line. Age of computers: game-based teaching of computer fundamentals. in Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education 2004. Leeds, United Kingdom ACM Press, 2004.

[7] Leutenegger, S. and J. Edgington, A games first approach to teaching introductory programming. SIGCSE Bull., 2007. **39**(1): p. 115-118.

[8] Singh, J.S.s.o.R. Learning Computer Programming Using A Board Game – Case Study on C-Jump. in MMU International Symposium on Information and Communications Technologies (M2USIC 2007), 2007.

[9] Hamid, S.H.A. and Y.F. Leong, Learn Programming by Using Mobile Edutainment Game Approach, in Proceedings of the The First IEEE International Workshop on Digital Game and Intelligent Toy Enhanced Learning 2007, IEEE Computer Society, 2007.

[10] Tan, P.H., S.W. Ling, and C.Y. Ting. A Review on Game Design and Game-based Learning Models. in UiTM International Conference on E-Learning (UICEL), 2007. Universiti Teknologi MARA, Shah Alam, Malaysia.

[11] Tan, P.H., S.W. Ling, and C.Y. Ting. Adaptive Digital Game-Based Learning Framework. in DIMEA 2007: Second International Conference on Digital Interactive Media in Entertainment and Arts, Perth, Western Australia: ACM Computers in Entertainment, 2007.