SEBHA UNIVERSITY JOURNAL OF PURE & APPLIED SCIENCES VOL.21 NO. 4 2022
DOI: 10.51984/JOPAS.V21I4.2258

مجلة جامعة سبها للعلوم البحتة والتطبيقية
Sebha University Journal of Pure & Applied Sciences

Journal homepage: www.sebhau.edu.ly/journal/index.php/jopas

Identifying the Difficulties of Learning Programming for Non-English Speakers at CQUniversity and Sebha University

*Ibrahim Nnass[1], Michael A. Cowling[2], Roger Hadgraft[3]

[1]College of Information Technology, Sebha University, Sebha, Libya
[2]Mobile Computing & Apps CQUniversity, School of Engineering and Technology, 160 Ann St Brisbane, Queensland 4000, Australia 2
[3]Educational Innovation & Research, Faculty of Engineering & Information Technology, University of Technology Sydney, Broadway, NSW 2007, Australia

Keywords:
Computer programming
learning programming
programming language
programming student
novice programmers

ABSTRACT

Since computers have become widely used, programming has become a critical skill. Programming languages are built upon English language words and phrases. It is possible that this could make learning an English-based programming language for non-English language speakers especially challenging. In actuality, the literature did not say much. While many focused on solutions to teaching programming, and some researchers focused on the problems that led to these solutions, very few researchers made any distinction between the language capabilities of novices. Based on the literature, this research study highlights issues that directly influence beginners learning programming, looking particularly at the difficulties faced by those that do not have English as a first language. The methodological approach used in this research is a mixed methods design, with the questionnaire method for data collection in both CQUniversity in Australia and Sebha University in Libya, with experienced programmers and novice programmers in various stages of their study. These data were categorized and analysed to identify areas of difficulty. It became clear that there was a difference. While Australian students identified issues with loop statements (Do...While) and other program logic, Libyan students identified the major problem to be error messages and their interpretation, with over 71% of students identifying this as a problem. However, error messages were mentioned by just 2% of participants at CQUniversity. It was clear that English was a problem.

تحديد صعوبات تعلم البرمجة للناطقين بغير اللغة الإنجليزية في جامعة كوينزلاند المركزية وجامعة سبها

*إبراهيم النعاس[1] و مايكل أ.كاولينج[2] و روجر هادجرافت[3]

[1]كلية تقنية المعلومات ، جامعة سبها ، سبها ، ليبيا
[2]الحوسبة المتنقلة والتطبيقات *CQUniversity*، كلية الهندسة والتكنولوجيا ، 160 Ann St Brisbane ، كوينزلاند 4000 ، أستراليا
[3] الابتكار التربوي والبحث، كلية الهندسة وتكنولوجيا المعلومات، جامعة التكنولوجيا سيدني، رودواي، نيو ساوث ويلز 2007، أستراليا

الكلمات المفتاحية:
برمجة الحاسوب
تعلم البرمجة
لغة برمجة
طالب البرمجة
المبرمجين المبتدئين

الملخص

منذ ان انتشر استخدام أجهزة الكمبيوتر على نطاق واسع، اصبحت البرمجة مهارة بالغة الأهمية. فلغات البرمجة مبنية على كلمات وعبارات اللغة الإنجليزية، وهذا جعل تعلم البرمجة يعتمد على اللغة الإنجليزية وبالتالي جعل تعلمها للغير الناطقين باللغة الإنجليزية أمرًا صعبًا بشكل خاص. الدراسات السابقة لم تقدم الكثير في هذا الجانب. ففي الدراسات السابقة ركز الكثيرون على حلول لتدريس البرمجة، البعض ركز على المشكلات التي أدت إلى هذه الحلول، بينما عدد قليل جدًا من الباحثين تطرق للقدرات اللغوية للمبتدئين. استنادًا إلى الدراسات السابقة، هذه الدراسة البحثية سوف تسلط الضوء على القضايا التي تؤثر بشكل مباشر على تعلم المبتدئين للبرمجة، وتبحث بشكل خاص في الصعوبات التي يواجهها أولئك الذين تعتبر اللغة الإنجليزية لديهم ليست لغة

Corresponding author:
E-mail addresses: Alnnass76@yahoo.com ,(M. A. Cowling) m.cowling@cqu.edu.au , (R. Hadgraft) Roger.Hadgraft@uts.edu.au
Article History: Received 30 May 2022 - Received in revised form 17 July 2022 - Accepted 03 October 2022

أولى. المنهجية المستخدمة في هذا البحث هي mixed methods ، تم تجميع البيانات باستخدام الاستبيانات في كل من جامعة كوينزلاند المركزية في أستراليا وجامعة سبها في ليبيا ، استهدفت المبرمجين ذوي خبرة والمبرمجين المبتدئين في مراحل مختلفة من دراستهم. تم تصنيف هذه البيانات وتحليلها لتحديد مجالات الصعوبة. نتيجة تحليل هذه الاستبيانات ثبت ان هناك فرقا واضحا، حيث حدد الطلاب الأستراليون المشكلات المتعلقة بالحلقات التكرارية ومنطق البرمجة، بينما حدد الطلاب بجامعة سبها المشكلة الرئيسية التي تواجههم على أنها رسائل الخطأ من حيث فهمها وكيفية التعامل معها، حيث حدد أكثر من 71٪ من الطلاب في المشاركين بجامعة سبها هذه المشكلة على أنها المشكلة الرئيسية. ومع ذلك، نسبة 2 ٪ فقط من المشاركين في جامعة كوينزلاند المركزية ذكروا ان لديهم مشكلة مع رسائل الخطأ. لذا اصبح من الواضح أن اللغة الإنجليزية مشكلة اساسية تواجه الطلاب بجامعة سبها.

Introduction

Computer programming is a predominantly English-based medium. Languages like C++ and Java all have their base in Silicon Valley, growing out of companies like Bell Labs and Sun Microsystems in the USA. This base is very technical and privileges the experienced computer scientist and computer engineer, rather than providing a platform to assist the beginner. This makes learning a programming language for novices, without any prior experience, especially challenging (Bonar & Soloway 1983). Novice student programmers at the beginning have much to learn in the first programming class such as how to write new code with appropriate formatting, correct syntax, logical sequence of steps, dealing with compiler error messages, and so on. Then, starting with modularized functionality and complex concepts for example arrays and pointers.

However, Pillay (2003) states that computer science students, when learning to write procedural and object-oriented programs for the first time, often face difficulties. Similarly, Tan et al. (2014) found that many of the novice programmers have faced several difficulties in understanding and mastering the skills of computer programming, noting that programming is considered more of a mental skill than other knowledge. Shuhidan, Hamilton and D'Souza (2009) discovered that the struggle to learn and understand programming is largely responsible for high attrition rates in computer science schools. Hu, Winikoff and Cranefield (2012, p. 43) found that 'introductory programming courses have been continuously reported as having a high rate of failure or withdrawal' Winslow (1996) suggests reasons for this phenomenon including lack of problem-solving strategies and plans, and lack of detailed mental models. Lee and Ko (2011) make the interesting point that the first line of novices' programs predominantly lead to unexpected mistakes, such as runtime errors, syntax errors, or program output that the novice did not intend and that 'all of these forms of feedback are essential to helping a beginner understand what programs are and how computers interpret them' (p. 109). Stefik and Siebert (2013, p. 1) state that 'recent studies in the literature have shown that syntax remains a significant barrier to novice computer science students in the field'. Winslow (1996, p.17) makes the claim 'that novice programmers know the syntax and semantics of individual statements, but they do not know how to combine these features into valid programs'.

Soloway (1986) indicates that teaching the syntax is not enough, and more is needed:

> 'Students should be given explicit instruction in "vocabulary terms" such as mechanism, explanation, goal, plan, rules of programming discourse, plan composition methods, etc' (Soloway 1986, p. 858).

Rongfang (2011) illustrates several problems facing novice programmers in understanding programming concepts and the futility of the learning methods used, which leads to a claim that:

> 'The education quality of computer programming language course to the undergraduates is significantly declining. The overall evaluation of graduates majored in computer science from business circles is that the number of applicants for employment is large, but fewer graduates can code skillfully. Even fewer can develop software'. (Rongfang 2011, p. 1267)

Black (2006, p. 103) indicates that 'learning how to write program code is similar to learning a foreign language'. Thus, Black (2006, p. 113) concludes that:

> 'Students in computer classes need to hear the explanation from the teacher and observe the demonstration skill or technique, but also to have time with the teacher for practice, feedback and discovery'.

Most programming languages use English keywords (while, for, if, and so on.), which are familiar for regular users of English, but unfamiliar for non-English language speakers. With this in mind, it seems possible that learning a programming language could be more difficult for non- English speakers. In countries such as Libya, programming faculty at universities and higher institutes are required to translate the programming courses from English into Arabic, the only language spoken in Libya. Novice programmers are forced to deal with English-based programming languages with only a basic understanding of the English language due to their schooling. Al-Hussein (2014) explains the reasons why English language teaching was stopped in Libya:

> Due to some political reasons, and after the air-raid against the Gaddafi regime led by the United States of America and the United Kingdom in mid of April 1986, and as a consequence, the Minister of Education made the decision No. 195/1986 to stop teaching foreign languages in Libya. The decision badly affected the future of education in Libya. Till the time of writing this material; very serious consequences relating to that decision are still active. (Al-Hussein 2014, p. 59)

This could make understanding an English-based programming language for Libyan novice programmers especially challenging. Looking past the difficulties with learning programming specifically and instead focussing on the language differences, the literature indicates that Arab students (especially Libyans) face a real problem in understanding the English language. Personally, as an undergraduate student in Libya I struggled to learn programming and I experienced many difficulties in understanding how to build a program correctly and then follow up its implementation; those difficulties were varied depending on the programming language used, for example, finding bugs, and understanding programming structures or how to track any code. So, based on my experience as a student, I understand the suffering of any student facing a similar case. This research paper is one part of a larger research project looking at the difficulties of learning programming for non-native English speakers. Specifically, this paper presents the results of the first phase of this project, and is an attempt to highlight the most important difficulties faced by novice programmers in learning computer programming in general and for non-native English speakers in particular. This paper in particular seeks to answer these research questions:

RQ1.0 What are the problems and difficulties that have a significant impact on non-English speakers at Sebha University in Libya in learning computer programming?

RQ1.1 What are the problems and difficulties that non-English speakers face in learning computer programming in general?

RQ1.2 What are the problems and difficulties that have a significant impact on English speakers at CQUniversity in Australia in learning computer programming?

RQ1.3 What are the differences between English speakers at CQUniversity and non-English speakers at Sebha University with respect to learning computer programming?

This paper conducted investigation through literature, as well as studies with programming students at the Australian and the Libyan university, to highlight the problems and difficulties that have a significant impact on programming novices. The expected outcome of this paper is a better understanding of the challenges of learning computer programming for both English-speaking novice programmers and non-native English speaking novice programmers.

LITERATURE REVIEW

This section provides a critical literature review of the difficulties that novices face in learning a programming language, especially for non-speakers of English. Some papers only deal with programming issues in the general sense, rather than highlighting specific problems (for example, Hadjerrouit (2008), Yinnan and Chaosheng (2012) and Kelleher and Pausch (2005)). Nonetheless, some of these papers provided useful lessons. Tie et al. (2012) concludes that the amount of time spent by beginners practising programming in homework plays a very important role in perfecting their programming language skills. Yinnan et al. (2012) confirms that it is difficult for beginners to master any programming language in a short period of time because programming languages have practical techniques. Hadjerrouit (2008), too, notes that it may be difficult for beginners to gain programming skills in a short time. In fact, Soloway and Spohrer (1989) concluded that novices need 10 years to become expert programmers. Kelleher and Pausch (2005) focus on some issues related to the learning of programming for beginners, for example, novices not seeing the relevance of programming or novices needing to feel they can achieve progress in learning programming. Hook and Eckerdal (2015) analysed the final exam results in an introductory programming course and then concluded that students who spent more time on the computer practicing programming problems had higher marks in the course.

Similarly, students who spent less time on practicing programming problems, but attended lectures and spent more time reading books, achieved lower marks. This finding led them to conclude that it is very important to encourage novice programmers to spend more time practising programming. Winslow (1996, p. 17) states that 'novice programmers know the syntax and semantics of individual statements, but they do not know how to combine these features into valid programs'. Al-Imamy and Alizadeh (2006) point out that the programming student spends a long time on the syntax of the language, which leaves little time to develop skills in software design and creative solutions due to students having different backgrounds, a teaching strategy that is traditional, and course duration linked to limited time. Piteira and Costa (2012), indicate that these problems often lead to high failure rates in first programming courses where time is limited to the academic semester. Tan et al. (2014) note that computer programming is considered more of a mental skill than other knowledge, and many of the novice programmers have faced several difficulties in understanding and mastering the skills of programming. Casey (1997) concludes that learning programming is full of challenges in terms of how to learn and practise problem-solving skills and other thinking skills and proposed that learning these skills should be part of any process for learning programming.

Casey goes on to suggest that the educational system has the main responsibilities surrounding programming: imparting knowledge, teaching cognitive skills, and teaching the most important skill, problem solving. Therefore, programming teachers must have a full understanding of programming and the problem-solving process. Although Casey confirmed the existence of the challenges in the field of teaching and learning programming, he did not specify the problems specifically or to find practical solutions for them; for example, he/she did not refer to the problem of English language and the language skills required to learn programming and problem-solving. Golding, Donaldson and Tennant (2009) note that there are many universities using innovative practices to develop students' performance in programming languages. However, learning programming continues to be difficult and students struggle with it. Thus, their research indicates that it is difficult for the teacher to help students to develop their abilities to learn and understand problem solving techniques and thus understand the programming language used. As mentioned above, almost all of these studies and research, which discussed the problems of learning programming, did not explicitly identify the programming issues that face the beginner non-speaker of English. These papers identified only the key general issues in learning programming. However, even with all these practices, some students still fail. In addition to these general papers, some papers also looked at specific issues which students faced.

1- Specific programming issues

There is literature that describes specific problems faced by students in general when learning programming. However, there is a lack of significant literature specifically focusing on the problems associated with learning programming for non-English speakers, with most papers focusing on the solutions. For example, Butler and Morgan (2007) describe a survey of the study habits and challenges faced by novice programmers. This paper focuses on the difficulties experienced by novices in understanding and implementing low-level programming concepts, like syntax and variables, and high-level concepts, for example, object-oriented programming principles and program design. The authors indicate that the most difficult issues faced by novice programmers are: algorithms, methods, object-oriented concepts, overall program and object design. Lahtinen et al. (2005) indicate that novices have several problems related to programming concepts and program construction, for instance, how to design a program to solve a certain task, how to divide functionality into procedures and how to find bugs in their own programs. These researchers indicate that the most difficult issues in programming concepts are: recursion, pointers and references, abstract data types and error handling and using the language libraries. In addition, they suggest that there are many novice programmers who have learning difficulties for reasons such as nature of the programming language, a lack of resources or a lack of personal instruction and novice groups that are large and heterogeneous. Xinogalos (2012) also indicates student difficulties, the most important of which are: developing an algorithm for solving a problem, transferring the algorithm to the programming language, dividing functionalities in functions or classes, understanding compilation error messages and correcting and finding bugs.

In addition, Milne and Rowe (2002) indicate that the most difficult topics depended on pointers and memory-related concepts (virtual functions, dynamic allocation of memory and polymorphism) because the novice programmers struggle to understand what is happening to their program in memory. Piteira and Costa (2012, 2013) indicate that the most difficult issues in programming concepts from the student viewpoint for computer programming topics are object-oriented concepts, overall program and object design, selection structures and variables (lifetime/scope), and loop structures. Also included are operators and precedence, structured data types and abstract data types, recursion and parameters, pointers and references, passing parameters, error handling and using the language libraries. Schulte and Bennedsen (2006) conclude that the difficult topics from a teacher's viewpoint include programming issues such as, algorithm efficiency, polymorphism and inheritance, generics (templates, type parameterisation). In addition, other programming difficulties included advanced data-structures (linked-lists, trees,…), design of classes (given a problem, determining the pre-defined classes needed to solve the problem), and divide and conquer (decomposition of a problem). Piteira and Costa (2013) support the findings of Lahtinen et al. (2005) and added two difficulties in programming: understanding programming structures, and learning the programming language syntax. Golding et al. (2009) highlight some important skills required for programming: problem solving skills, logical reasoning and mathematical thinking capabilities, self-efficacy, previous programming experience, mental models, coding, debugging, abstraction, and knowledge of syntax. The researchers state that there many universities are using innovative practices to develop skills and enhance students' performance in programming languages. However, even with all these practices, some students still fail. In summary, these papers focused specifically on programming issues that face novice programmers in general. These issues create difficulties and

challenges that lead to high drop-out rates on programming courses. Problem solving skills, logical reasoning, mathematical thinking capabilities and knowledge of syntax are important skills required to learn programming for novices (English speakers and non-English speakers). Yet, all papers reviewed indicate that the most difficult issues in programming concepts are recursion, pointers, references, using the language libraries, developing an algorithm for solving a problem, transferring the algorithm to the programming language, dividing functionalities in functions or classes, understanding compilation error messages and correcting, finding bugs, object oriented concepts, loop structures, operators and precedence, polymorphism and inheritance. All these concepts, however, need to be examined with non-English speaker programmers in mind.

2- English language issues

However, data was still missing on non-English speaker programming issues. Hence, a further literature review was conducted in this area, but only a few papers were found, as below. An example is Mhashi and Alakeel (2013), who conducted an investigation and analysis of the problems faced by novices, with two main goals in mind:

- To determine whether the novices at their university faced problems in computer programming similar to those faced by the novices in different universities around the world.
- To study the effect of sociocultural and environmental factors on learning computer programming skills.

Iqbal and Coldwell (2017) conducted a learning study in Buraimi University College, Oman (non-English speaking country). Iqbal and Coldwell (2017, p.790) indicate that 'instructors deliver the lectures and conduct practical sessions. The programming examples and exercises discussed during the lectures, and practical sessions promote a shortcut approach to programming, where the problem statement is directly converted into a computer program'.

At this institution, according to these researchers, 'instructors deliver the lectures and conduct practical sessions. The programming examples and exercises discussed during the lectures, and practical sessions promote a shortcut approach to programming, where the problem statement is directly converted into a computer program' (Iqbal and Coldwell 2017, p. 790). They go on to state that the main challenge when novices start in learning programming is that a number of different groups of skills must be learnt at the same time. However, the traditional approach used in teaching an introductory programming course does not provide students with all the required skills to understand programming (Iqbal and Coldwell 2017). Iqbal and Harsh (2013) suggest teaching programming through traditional approaches emphasise the syntax and semantics of the programming language, rather than problem-solving skills, in order to handle programming problems. Thus, novice programmers not only must learn the syntax and semantics of a programming language but also must develop appropriate problem-solving strategies.

Iqbal and Coldwell (2016) applied ADRI (Approach, Deployment, Result, Improvement) in the teaching and learning process of an introductory programming course to improve learning and success rates. The ADRI approach covers three types of programming errors: syntax errors, semantic errors, and syntax warnings. Their investigation found that more attention should be given to syntax errors and to semantic errors in the teaching process. Also, they discovered that repetition structures (loops), functions, and arrays were significant as they were found to be the most difficult topics in learning programming.

In the traditional approach there is one significant challenge for the novices to understand the hidden process going on in the computer memory during the compilation of the program. Particularly, novices do not understand the flow of information in the program, which leads to lack of understanding of the problem scope (Iqbal and Coldwell 2016). In the traditional approach the one significant challenge for the novices to understand the hidden process going on in the computer memory during the compilation of the program, is not understanding the flow of information, which leads to a lack of understanding of the problem scope (Iqbal and Coldwell 2016). Al-Nuaim, Allinjawi, Krause, and Tang (2011) found that many novice student programmers in computer science courses start without appropriate problem-solving skills or logical tools needed for problem-solving or start with a weak understanding of the simple mathematical requirements. Novices tend to dedicate their energies to learning syntax rather than learning the basic concepts of the scope. Therefore, beginners depend on trial and error rather than learning how to acquire problem solving skills. In addition, many novice programmers have a limited ability to carry out simple programming tasks (Al-Nuaim, Allinjawi, Krause, and Tang 2011).

Al-Nuaim, Allinjawi, Krause and Tang (2011) also note that the high schools in the Kingdom of Saudi Arabia devote inadequate time to teaching the English language. The paper further suggests that the curricula at these high schools lack the teaching of problem-solving skills in computer science courses as computer science is a completely new field to them.

3- Literature gap

In the literature review, it was thus identified that a set of concepts are difficult for programming novices, and that these concepts should have high priority for finding appropriate solutions. However, it was also discovered during the literature review that few studies have focused on learning programming for non-English speakers, specifically, as it related to the syntax of the programming language. However, the literature review did help to answer the following research question:

RQ1.1 What are the problems and difficulties that non-English speakers face in learning computer programming in general?

To address all of these challenges from the literature can be difficult, especially for novice programmers. Also, novices need to understand the program code quite comprehensively if they genuinely wish to be able to handle any program error. In addition, a novice needs to be able to search the information that will help him/her to use the language libraries, which is a very difficult step for novices.

But, perhaps the most interesting finding is that, in summary, most previous studies have targeted English speakers. Therefore, the gap in the literature that led to the research questions is that all these references do not refer to the problem of English language and the language skills required to learn programming. In addition, there were no previous or current studies referencing the problem of the English language in this context.

Thus, the researcher decided to conduct more investigation through a survey of students at CQUniversity in Australia and Sebha University in Libya to highlight the problems and difficulties that have a significant impact on novice programmers where language is a factor. Then, the knowledge gap identified through the literature review will be addressed. Thus, it became necessary to design a theoretical framework for this study to collect data from English language speakers and non- English language speakers.

Methodology & Research design

To work towards improving the current research methodological approach, a mixed methods design was selected for this research, combined with concurrent triangulation as a methodological strategy, with the questionnaire method as the data collection method rather than interviews as the best available solution in both CQUniversity and Sebha University. Thus, this research selected a mixed methods research (MMR) approach, which would include both qualitative and quantitative methods. The specific method type that is used through this research is triangulation design (as a methodological approach). According to Creswell (2009, p. 14) 'the concept of mixing methods originated in 1959 when Campbell and Fisk used multi-methods to study the validity of psychological traits'. The objective of using this design is 'to obtain different but complementary data on the same topic' (Morse, 1991, p. 122) to best understand the research problem, because this is important for the researcher to study and deeply understand all aspects surrounding the problem to find the most appropriate way to address it. Johnson and Onwuegbuzie (2004, p. 17) define mixed methods research as where 'the researcher mixes or combines quantitative and qualitative research techniques, methods, approaches, concepts or language into a single study'. In order to understand the difficulties that have a significant impact on novices at an Australian University in Australia and at a Libyan University from a staff perspective, the staff survey was designed and conducted with faculty members at the Australian university and at the Libyan

university who taught an introduction to programming course in Term 2, 2015 and Term 1, 2016. Therefore, the surveys were a practical approach. Triangulation as a methodological approach had one data collection method which is questionnaires. The questionnaires method was used to study programming issues as they are; it provided an accurate description of the programming issues for programming novices and clarified their characteristics.

The Java courses at both universities are similar with few differences. The reasons behind selecting Java courses, in particular, was because the main target groups in this study are undergraduate students (novice programmers), who are studying programming for the first time without any prior background on programming.

Theoretical Framework

After identifying the gap in the literature review, this research has been a process of constructing a theoretical framework for the collection of data collection from English language speakers and non-English language speakers. literature review includes programming concepts that have a significant impact on novices according to the literature. This framework is outlined in Figure 1, together with the match with programming concepts in literature review. The framework for this research was designed around Bloom's Taxonomy.

The most important reason for using Bloom's Taxonomy is: Bloom's Taxonomy is based on a logical premise that each level must be mastered before moving towards the other level (levels arranged according to the level of difficulty from least to highest). Thus, the levels within Bloom's Taxonomy are levels of educational development increasing in difficulty (Bloom 1956). In order for non-English-speaking students (novice programmers) to learn programming based on Bloom's Taxonomy, they must be familiar with the minimum of English so that they can begin to learn programming principles, ranging from easy to more complex concepts.

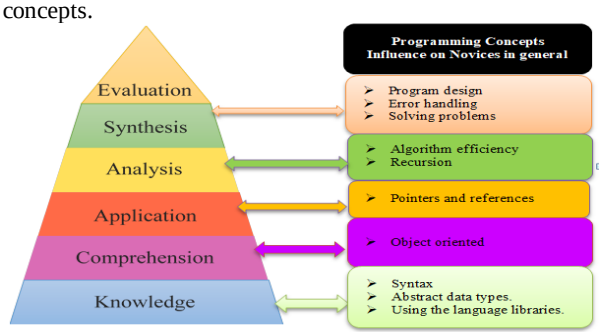

Fig. 1: Theoretical framework of research

Understanding Student Difficulties

This section explores the difficulties that novices have in learning programming. It includes the survey results of novice programmers from both universities (CQUniversity and Sebha University), contrasting those with a strong English background with those who have little to no English. The chapter also includes a summary of faculty members' feedback at CQUniversity and Sebha University.

1- Results from CQUniversity

One hundred and three (103) novice programmers participated in these surveys, and the main demographics are:

- **73** participants were without any prior experience.
- **21** participants did not indicate if they had prior experience or not.
- **4** participants were redoing this course and they did not indicate any issues in understanding or face any difficulties related to programming concepts.
- **5** participants did not indicate any issues in understanding or face any difficulties related to programming concepts.

Table 1 contains specific programming issues faced by novice programmers at CQUniversity.

Table 1: Specific programming issues faced by novice programmers at CQUniversity.

Understanding issues	No. of participants
Loop statements (Do. While)	16
All programming concepts	16
Program design	14
No issues	13
Syntax	11
Programming logic	6
Array and Manipulating arrays of objects	5
Some difficulties with programming concepts	4
Semantics	4
Data types	3
Object oriented programming	2
Error message	2
Condition statements	1

Finally, lists, schemes and categories were formulated as follows:

- Understanding the complicated technical language
- A complete lack of experience
- How to apply what is learned in class and how to explain my understanding
- Novices who see the fault as one of course design
- Lack of necessary skills
- Mainly time free of distractions (outside influences)
- Hard time remembering things.

2- Results from Sebha University

By contrast, seventy-two (72) novice programmers participated in the survey conducted at Sebha University, and the main demographics were:

- **39** participants were without any prior experience and doing this course for the first time.
- **33** participants indicated that they had prior experience and they are doing this course for the first time. And one of them did not indicate any issues in understanding or face any difficulties related to programming concepts.

The analysis that was undertaken to identify these specific programming issues is the qualitative content analysis process. This included transcribing data, coding, and categorising it. Then, by reading the text closely, the text was organised for analysis and units for analysis were determined. The text was checked again, looking for evidence of expected and unexpected categories. Finally, lists, categories and schemes were developed, which organised information and showed the relationships between them.

Table 2: Specific programming issues faces novice programmers at Sebha University

Understanding issues	No. of participants
Error messages	52
Pointers	34
Loop statements (Do. While)	29
All programming concepts	17
Array	14
Program design	11
Condition statements	8
Syntax	7
Semantics	1
Data types	1
No issues	1

Participants' comments and feedback were collected and categorised. Then, by reading the text closely, the text was organised for analysis and units for analysis were determined. The text was checked again, looking for evidence of expected and unexpected categories. Lists, schemes and categories that emerged following that were:

- A novice who sees the fault as one of course design such as: Lecturer should make more effort to teach programming step by step, lecturer should explain programming topics in a simple way and without ambiguity, increase tutorial time, stimulate the imagination of students' creativity, explain and simplify programs and dealing with students as beginners.
- Required skills related to learning programming languages: Learning, thinking and research skills, improving typing skills to save time, developing students' skills in the English language, program design skills and how to write a program correctly, problem-solving skills.

In summary, the results of the Sebha University survey displayed in Table 2, which contains specific programming issues faced by novice programmers at Sebha University in Term 1, 2016, revealed that there was a difference between these Libyan students and the

CQUniversity students, and the researcher identified one problem (error messages) that was particularly influential on the ability of novice programmers to learn programming language at the main Sebha University campus. Two other issues of importance for these students were pointers and loop statements (Do…While).

Error messages being indicated by students so frequently as an issue is likely not surprising given that each error message, in English, must be interpreted by someone with limited English ability. Error messages should, therefore, have the highest priority for addressing and finding appropriate solutions or developing methods or skills that may help novice programmers to learn programming.

In contrast, the results from CQUniversity (Table 1) suggest that the main issues for novice programmers in learning programming are loop statements (Do...While), all programming concepts and program design.

3- Updating the theoretical framework

Through this research study the theoretical framework was also updated to show which programming issue has the greatest influence (error messages) on non-English speakers at Sebha University.

Based on results from previous studies and results from CQUniversity and Sebha University, the researcher concluded that novices must master basic programming concepts in the knowledge level of Bloom's Taxonomy such as syntax, abstract data types and using the language libraries before moving to the advanced programming concepts at higher levels of Bloom's Taxonomy. It is clear that many students remain stuck on these lower levels, related to the syntax and concepts of programming. Thus, students at Sebha University have a lack of understanding of the English language that can lead to a lack of comprehension of error messages in the synthesis level of Bloom's Taxonomy. The feedback from participants (undergraduate students) at Sebha University supporting this point includes: practising programming frequently to gain experience, developing students' skills in the English language and developing learning, thinking and research skills. Bloom's Taxonomy attempts to divide the cognitive domain into subdivisions extending from the simplest to the most complex. So, error messages are in the most complex levels of Bloom's Taxonomy (analysis, synthesis and evaluation), for example, Java error "Possible loss of precision". It means that the student did something like K = H; where K is an int and H is a double. The novice must have the ability to read and understand error messages, then, to determine the exact location of the error in the program and then process and correct the error. This will not occur if the novice lacks the ability to read and understand the error message. Furthermore, novices may face difficulties in understanding basic programming concepts at the lower levels of the Bloom classification. Thus, all these difficulties will make learning and understanding error messages extremely challenging for novices, especially for Sebha University students. **Figure 2** contains an updated theoretical framework for this research study. The updated theoretical framework presents clearly the programming issue that has the greatest influence on novice programmers (non-English speakers) at Sebha University, which are error messages (System Errors, Syntax Errors and Semantic Errors).

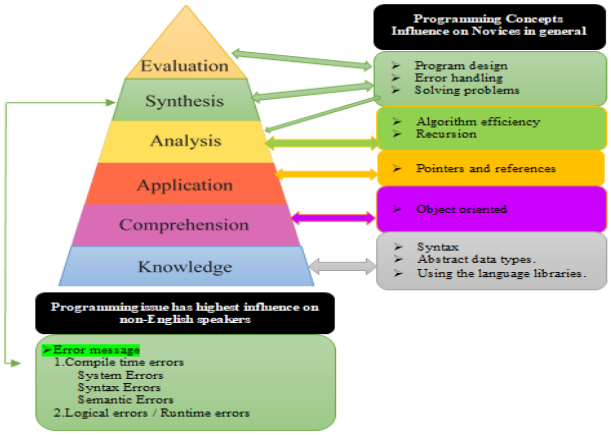

Fig. 2: Updated Theoretical framework of research

So, to deal with and address all of these challenges discovered in the literature, which face programming novices at the same time, it needs to be acknowledged that learning and understanding programming by novice programmers is difficult. Also, novices need to understand the program code comprehensively if they genuinely wish to be able to handle any program error. In addition, novices need to independently search the information that will help them use the language libraries, which is a very difficult step for novices. Thus, with the results from CQUniversity and Sebha University, this provides an answer to research questions RQ 1.0, RQ 1.2 and RQ 1.3, and lays the foundation for future work with non-English speakers. Based on the literature review, the results from the previous study, as well as these new results, it is clear that the problems faced by students in the surveys at the two universities were different.

Thus, these concepts that were identified in survey results at Sebha University and CQUniversity should have high priority for addressing and finding appropriate solutions. The ultimate purpose behind conducting these surveys is to study in depth these programming issues from both previous studies and through collecting and analysing data from CQUniversity and Sebha University, then develop a method to help novices in general and non-English speakers in particular. Any planned intervention at Sebha University should be targeted towards the biggest problem faced by those students, which is interpreting error messages.

In conclusion, this research study contributes new knowledge to the literature and practice, providing insight that has not been researched before, namely problems faced by novice programmers who are non-English speakers.

Conclusion

Writing and debugging programs can be very difficult, especially for novice programmers. Novices need to be able to search the language libraries and understand the program code comprehensively if they genuinely wish to be able to debug their programs and avoid/reduce error messages, which is a very difficult step for novices particularly as it is English language intensive. The ultimate purpose behind conducting this research was to investigate these challenges faced by novice programmers in general and non-English speakers in particular. In summary, this research study examined the difficulties and challenges faced by programming novices at both CQUniversity and Sebha University. The results from surveys conducted at both universities were different. The results from Sebha University are interesting in that the main issues were error messages, pointers, and loop statements (Do...While). However, the results from CQUniversity suggest that the main issues that faced novice programmers when learning programming were loop statements (Do...While), all programming concepts and program design.

Perhaps the most interesting finding is that error messages have the largest impact on novice programmers at Sebha University and should therefore have high priority for finding appropriate solutions. These programming issues must be studied in-depth and understood so as to design the most suitable solution. The qualitative data at Sebha University points out that one of the key areas of concern for Sebha University participants is one of course design, for example, the lecturer should make more effort to teach debugging and explain programming topics step by step in a simple way and without ambiguity, specifically, what do error messages mean and how to deal with them, and to increase tutorial time. (Maybe they also need to teach debugging – specifically, what do error messages mean and how to deal with them). Also, participants pointed out some required skills related to learning programming languages, for instance, practising programming frequently to gain experience, developing students' skills in the English language and improving learning, thinking and research skills. However, this research has some limitations, like all research, for instance, the research intervention through this study was also geographically limited to one Australian University and one Libyan University. That may have led to limitations with regards to the implication of findings or improving solutions based on identified problems for other universities or other countries.

References

[1] Al-Hussein, S. M. (2014), Teaching English as a Foreign Language in Libya. Scientific Research Journal (SCIRJ). 58-64.

[2] Al-Imamy, S., and Alizadeh, J. (2006), On the development programming teaching tool: the effect of teaching by templat the learning process. Journal of Information Technology Education. 5. 271-283. http://www.jite.org/documents/Vol5/v5p271-283Al-Imamy115.pdf

[3] Al-Nuaim, H, Allinjawi, A, Krause, P, & Tang, L 2011, 'Diagnosing student learning problems in object oriented programming', Computer Technology and Application, vol. pp. 858–865.

[4] Bloom, BS 1956, Taxonomy of educational objectives: the classification of educational goals, New York: Longmans, Green.

[5] Bonar, J & Soloway, E 1983, 'Uncovering principles of novice programming'. Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on principles of programming languages, pp.10–13.

[6] Butler, M., and Morgan, M. (2007), Learning challenges faced by novice programming students studying high level and low feedback concepts. Proceedings ascilite Singapore. 99-107.

[7] Casey, P. J. (1997), Computer programming: a medium for teaching problem solving. Computers in the Schools. 13. 1-2. 41-51.

[8] Creswell, JW. (2009), Research design: qualitative. quantitative and mixed methods approaches. Sage. Thousand Oaks. CA.

[9] Golding, P., Donaldson, O., and Tennant, V. (2009), Application of modified perceived learning Problem Inventory (PLPI) to investigate performance in introductory programming. IEEE Frontiers in Education Conference. 1-6.

[10] Hadjerrouit, S. (2008), Towards a blended learning model for teaching and learning computer programming: A case study. Informatics in Education-An International Journal. 7. 2. 181-210.

[11] Hook, LJ & Eckerdal, A 2015, 'On the bimodality in an introductory programming course', Proceedings of IEEE International Conference on Learning and Teaching in Computer and Engineering (LaTiCE), pp. 79-86.

[12] Hu, M, Winikoff, M & Cranefield, S 2012, 'Teaching novice programming using goals and plans in a visual notation', In Proceedings of the Fourteenth Australasian Computing Education Conference, vol. 123, pp. 43-52.

[13] Iqbal, S., & Coldwell-Neilson, J (2016), A model for teaching an introductory programming course using ADRI, Education and Information Technologies, vol. 22, no. 3, pp.1089-1120.

[14] Iqbal, S., & Coldwell-Neilson, J (2017), Impact of a New Teaching and Learning Approach in an Introductory Programming Course, Journal of Educational Computing Research, vol.55, no. 6, pp.789–819.

[15] Iqbal, S., & Harsh, O. K (2013), A self-review and external review model for teaching and assessing novice programmer International Journal of Information and Education Technology, vol.3, no.2, pp.120-123.

[16] Johnson, RB., and Onwuegbuzie, AJ. (2004), Mixed methods research: A research paradigm whose time has come. Educational Researchers. 33. 7. 14-26.

[17] Kelleher, C., and Pausch, R. (2005), Lowering the Barriers to Programming: A Taxonomy of Programming Environments Languages for Novice Programmers. ACM Computing Surve 37. 2. 83-137.

[18] Lahtinen, E., Ala-Mutka, K., and Järvinen, H. (2005), A stud of the difficulties of novice programmers. ACM SIGCSE Bulletin. 37. 3. 14-18.

[19] Lee, M & Ko, A 2011, 'Personifying programming tool feedl improves novice programmers' learning', Proceedings of the seventh international workshop on computing education rese pp.109–116.

[20] Mhashi, M. M., and Alakeel, A. (2013), Difficulties Facing Students in Learning Computer Programming Skills at Tabuk University. Recent Advances in Modern Educational Technologies. 15-24.

[21] Milne, I & Rowe, G 2002, 'Difficulties in learning and teachi programming—views of students and tutors', Education and Information technologies, vol. 7, no. 1, pp. 55-66.

[22] Morse, JM. (1991), 'Approaches to qualitative-quantitative methodological triangulation. Nursing Research. 40. 2. 120–

[23] Pillay, N 2003, 'Developing intelligent programming tutors f novice programmers', ACM SIGCSE Bulletin, vol. 35, no. 2 pp.78–82.

[24] Piteira, M., and Costa, C. (2012), Computer programming an novice programmers. Proceedings of the Workshop on Information Systems and Design of Communication ACM. 5 53.

[25] Piteira, M., and Costa, C. (2013), Learning computer programming: study of difficulties in learning programming. International Conference on Information Systems and Desigr Communication. 75-80.

[26] Schulte, C., and Bennedsen, J. (2006), What do teachers teac introductory programming?. ACM. 17-28.

[27] Shuhidan, S., Hamilton, M & D'Souza, D 2009, 'A taxonomi study of novice programming summative assessment', Proceedings of the Eleventh Australasian Conference on computing education, vol. 95, pp.147–156.

[28] Soloway, E & Spohrer, J 1989, Studying the novice program Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 497.

[29] Soloway, E 1986, 'Learning to program = learning to constru mechanisms and explanations', Communications of the The Association for Computing Machinery (ACM), vol. 29, no. 9 pp.850–858

[30] Stefik, A & Siebert, S 2013, 'An empirical investigation into programming language syntax', ACM Transactions on Computing Education, vol. 13, no. 4, pp. 1-40.

[31] Tan, J, Guo, X, Zheng, W & Zhong, M 2014, 'Case-based teaching using the laboratory animal system for learning C/C programming', Computers & Education, vol. 77, pp. 39-49

[32] Tie, Z., Zhuang, H., Zhang, Q., and Wang, Z. (2012), Analys the relationship between student grades and computer programming time in learning the C programming language. International Conference On Computer Science & Education (ICCSE). 1584-1589.

[33] Xinogalos, S. (2012), Programming techniques and environn in a technology management department. ACM. 136-141.

[34] Winslow, LE 1996, 'Programming pedagogy a psychological overview', ACM SIGCSE Bulletin, vol. 28, no. 3, pp. 17-22.

[35] Yinnan, Z., and Chaosheng, L. (2012), Training for computational thinking capability on programming language teaching. International Conference on Computer Science & Education (ICCSE). 1804-1809.