www.ijemr.net ISSN (ONLINE): 2250-0758, ISSN (PRINT): 2394-6962

Volume-8, Issue-3, June 2018

International Journal of Engineering and Management Research

Page Number: 191-194

DOI: doi.org/10.31033/ijemr.8.3.26

Teaching Programming to Non-Programmers at Undergraduate Level

Juena Ahmed Noshin[1] and Syed Ishteaque Ahmed[2]

[1]Lecturer, Department of Computer Science, American International University-Bangladesh, BANGLADESH

[2]Business Analyst Consultant Vice Chair, Business Analysis Community, PMI® Toronto Chapter, CANADA

[1]Corresponding Author: juena@aiub.edu

ABSTRACT

This paper focuses on solving the problems faced by non-IT (Information Technology) background students at undergraduate level in learning programming language who are at the same time non-native English language speakers. In this regard a step by step methodology has been proposed here which try to take into account the difficulties faced by this particular niche of programmers and counteract them with possible solutions. Following this approach may help lower the problems faced by the non-IT background students to some extent and fulfill their aim in being professional or conversational or end- user programmers according to their career choices.

Keywords--- Non-IT Background Students, Non-Native English Speakers, Programming, Undergraduate-Level Teaching

I. INTRODUCTION

A lot of focus has been given in teaching programming to IT background students. However, till now due to a variety of different reasons the same importance is not given to the teaching methodology of non-IT background students. But this tactic has to be changed as even in disciplines outside the IT domain coding has become a necessity. Graduates from other specialization specially engineering and business studies have started taking programming seriously as it enhances the weight of their resume to potential recruiters. Now-a-days in the job market programming language skill is compared equivalent to having English language skill. But it is not always necessary that all programmers will code as most programmers do not write a single line of code in their professional career. However, they spend a lot of time conversing with programmers. This gives rise to the need to understand programming language and have influenced the necessity of employees who are fluent in conversing with programmers. Another class of programmers is the end-user programmers who need to know programming for research or job-related work though they are not professional programmers. From these scenarios it is evident that programming should be given emphasis by even non-IT background students as it will help them take their career to an even more lucrative path after graduation. For these reasons a more concentrated teaching methodology dedicated to only non-IT background students should be implemented as it has become very important for all students to have skills in the programming domain irrespective of their educational expertise.

II. PRIOR APPROACH

Various researchers have thought about different techniques in teaching introductory programming courses to IT background students. However, not too much concentration has been given in teaching non-IT background students programming language. To some extent this scenario has started to change since the last decade. This change occurred mainly because of the rise of computer professionals and the necessity to communicate with them in daily basis. Though not many, but some unconventional research work has been done to help beginners learn programming which is discussed here.

Novice coders face a lot of problems when learning a programming language. This is because concepts like variables and loops are new and something that they do not have to think about while solving everyday problems. Moreover, trying to maintain grammatical aspects i.e. syntax leads to greater complexity and de-motivates beginners. Most of the new students solve coding problems based on trial and error rather than trying to perceive the underlying problem. Biswajit Saha

Copyright © 2018. IJEMR. All Rights Reserved.

and Utpal Kumar Ray, 2015found out that at an earlier stage of a programmer's life concentration should be given not only on coding but also on software engineering concepts like documentation, reusability, coding style etc. This will benefit the programmers in their professional career.

Non-IT background final year students at undergraduate level may need software and programming skill to help with their thesis and research work. So, what they require is a short and effective course which will help them learn the basics of programming. This is of course not an easy task as it is pretty time consuming even for an IT background student to learn programming and it may take them years to get a firm grasp of it. However, non-IT background students are not granted this luxury of infinite time. To solve this problem a hybrid approach combining the American and Chinese teaching practices has been adopted by Zhen Jiang et al., 2011 where concentration is given on teaching programming to non-programmers through the use of loops. To make the approach interesting, gaming with loops is introduced along with interactive one error solving problems that help non - programmers grasp the concepts easily. This approach has been quite effective and adopted by a number of reputed institution.

For novice learners generally, programming seems difficult irrespective of what major they are from. Almost all popular programming languages are English language based. This deems as an obstacle for non-native English speakers. On one hand there is difficulty in understanding various syntax of programming language. On the other hand, the pressure of understanding English vocabulary is added. Does these two problems combined together enhance the suffering of new learners is the main research question of Ashok Kumar Veerasamy and Anna Shillabeer, 2014. To find that out a conceptual model have been proposed that chooses important programming language keywords and ask student if they know the meaning in English context and also programming context. This helps to understand if there is direct correlation between English language efficiency and programming language efficiency.

Conventionally programming is taught to students in a grammar first approach. Rongfang Gao, 2011 focused on understanding the problem first and considering the grammatical aspect later. Firstly, a program is selected and by studying it in groups students try to enhance their knowledge rather than the teacher describing every aspect in details. The research has proved its efficiency through a four-year procedure followed in a reputed institution.

Sue Sentence and Andrew Csizmadia, 2015presented a teaching methodology of programming language to students from age group 5 to 16 years. Here school teachers have suggested strategies like pair programming, real life examples, algorithm, error tracing and physical activities as effective ways to teach beginners code writing.

A new class of programmers is the conversional programmers which have been proposed by Chilana et al., 2015 and Chilana et al., 2016. They are non-IT background programmers whose main aim is to be able to understand and communicate in programming language. The major incentive for these new era programmers is the value of this skill in job market. This conversing skill of potential employees can be compared equivalent to the skill of learning a foreign language to recruiters. Chilana et al., 2015 deals with the research question "Should the non-programmers, conversational programmers and end user programmers be taught programming the same way?" Moreover, from this research it has been concluded that introductory programming courses gave non-IT background students the understandability that coding is not typing but rather solving problems. Understanding this fact has garnered respect for IT professionals in the mind of the non-IT background students which is required as they have to work side by side throughout their professional career.

Im et al., 2017 incorporated music to make programming seem more interesting and exciting to non-IT background coders. In this regard, a musical learning platform was introduced which helped teach basics of programming like loops, variables etc. by incorporating music-based lab tasks. The approach significantly increased student interest in programming.

Chilana et al., 2016 identified that a lot of programmers now-a-days do not want to do coding professionally rather they want to learn coding so that they can converse with developers. This enhances their marketability to a great extent. However, a research question arises that should a new pedagogy be introduced where communication using programming jargon is taught rather than raw coding?

Most students have a fixed mindset about programming even before starting a course that it is very hard and boring. To get this notion out of their mindset Azad Ali and David Smith, 2014 used Alice as an introductory programming language. Alice is an animated language that does not give any syntactical error so the students can focus on the coding concepts rather than grammar. The advantage of using this sort of language is that students find it interesting and easy. But the disadvantage is that learning languages like Alice is not a marketable skill.

Dennis R. Goldenson, 1996 concluded that thinking skill and logical understanding of humans can be enhanced through learning of programming languages. The concept has been verified through the use of programming language Karel.

Though some research work has been done regarding teaching programming at introductory level and

Copyright © 2018. IJEMR. All Rights Reserved.

many effective approaches have been introduced by teachers and researchers alike, to our knowledge very few researches have been done giving full concentration on teaching programming to non-programmers at undergraduate level majoring in engineering who are non-native English speakers. Our research solely focuses on this problem and tries to propose an appropriate solution.

III. OUR APPROACH

The approach proposed here concentrates on non-IT background students who are majoring in engineering taking their first official programming course at undergraduate level. Only those students who are non-native English speakers are taken into consideration. To help them grasp the topics of programming some steps may be taken. The steps are listed below.

a. Choosing Appropriate Language

Students as a whole find it difficult to program initially. To solve this deficiency especially for the non-IT background undergraduate students', languages like Alice and Karel can be used. However, these languages almost have next to no value in the job field. So, after much consideration C was chosen as an appropriate language for teaching introductory programming to non-IT background students.

b. Subset of Original Syllabus

Since most non-IT background programmers take introductory coding courses to become conversational programmers it is a better approach to teach them basics rather than involving too much conceptual topics. In this regard a subset of the C syllabus of IT background students is taken as the syllabus for the non-IT background students. This tactic will enhance non-IT background student coding skill and gave some sorts of equilibrium with the IT background students in their knowledge level.

c. Kinesthetic Learning

From previous studies it is evident that physical hands-on training really helps students learn. This approach has already been tried at middle school level however we have tried to incorporate kinesthetic learning in undergraduate level also. It is evident that visual aids help student learn. For example, topics like loops can be taught involving student volunteers who physically demonstrate their usage with ropes.

d. Vocabulary Workshop

For this research only, non-native English language speakers have been considered. It is a challenge in itself for a non-native English speaker to understand English properly. To add the dynamic of a logical language adds to the trivia. So, it becomes near to impossible for new learners to grasp on the subject matter of programming language easily. To help this agenda a vocabulary workshop can be held to help to clear the English meaning of the C language keywords along with the meaning of the keyword in the programming context. This will help the students grasp the subject matter easily.

e. Problem Description

There is a common misconception in student community that programming is just typing. So, whenever they are given a task they start typing the syntax without further thinking. This practice should be strictly opposed from the beginning. Students shall be taught to first think and then start to code. In this regard the teachers will first discuss about the problem in detail and how solving it through programming will decrease human work load in the long run. This will motivate the students and also help them understand coding better.

f. Contextual Tasks

There is no alternative to solving lab exercises to understand the C topics better. However, if lab tasks are given based on the educational expertise of the students they find more enthusiasm in solving the problems. For example, if Electrical Science undergraduates are given exercises based on circuits then its more related to their subject matter hence they find it more interesting and easier to solve. This approach will help the students clarify that programming is needed in every field.

g. Pair Programming

Usually software engineering concepts are introduced much later on in an IT-background student's life i.e. usually on the third year. However, in this approach pair programming which is a concept of software engineering is introduced at introductory level as it will help students learn from each other. It will also develop the habit of conversing in programming jargon which most non-IT background students aim to do in their job life.

h. Learning to Troubleshoot

To understand a programming language deeply it is not enough to just learn how to code. A student must also learn to find errors in an existing code, know the output from seeing a given code without using compilers and even be able to finish an incomplete code if given. These practices will help them gain a deeper understanding of C language which will help them become better programmers later on in life.

i. Internal Documentation

From the very beginning of their programming education students should learn to put comments on their code. This approach will help them understand the code they have written even better and also help them in reusing the codes in future assignments. Moreover, internal documentation i.e. commenting helps other students understand the code which in turn aids in pair programming.

The steps mentioned above if followed properly will ensure that a novice programmer gain enough knowledge to survive in the ever-evolving industry which now considers programming language as a competitive skill near equivalent to speaking in English fluently.

Copyright © 2018. IJEMR. All Rights Reserved.

IV. CONCLUSION

In this modern era, it is considered a top-notch skill to be able to understand and converse in programming language. However, for non-IT background students it is not a very easy task to acquire this skill. To help these non-IT background introductory programmers some steps have been proposed in this paper which will aid them with overall learning process and in turn will help the students become great conversers and programmers in their professional careers.

REFERENCES

[1] Biswajit Saha & Utpal Kumar Ray. (2015).Learning programming: An Indianperspective. *International Journal of Information Science and Computing*, *2*(1), 21-32.

[2] Zhen Jiang, Eduardo B. Fernandez,& Liang Cheng (2011). A Pedagogical Pattern for Teaching Computer Programming to Non-CS Majors.*Proceedings of the 18th Conference on Pattern Languages of Programs*, Portland, Oregon, USA.1-10. Available at: https://pdfs.semanticscholar.org/ac30/b21adc7f3d3c131d5f38b6be168cc1230f5c.pdf

[3] Ashok Kumar Veerasamy & Anna Shillabeer. (2016). Teaching English Based Programming Courses to English Language Learners/Non-Native Speakers of English. *International conference on Education and Management Innovation*, Hong Kong. Available at: https://www.researchgate.net/publication/304024941_Teaching_English_based_programming_courses_to_English_learnersnon-native_speakers_of_English

[4] Rongfang Gao (2011). Reforming to Improve the Teaching Quality of Computer Programming Language. *6th International Conference on Computer Science & Education (ICCSE)*, Singapore.1267-1269.Available at: https://ieeexplore.ieee.org/document/6028863/

[5] Sue Sentence & Andrew Csizmadia. (2015). Teachers' perspectives on successful strategies for teaching Computing in school. *IFIP TC3*, *Vilnius, Lithuania*,1-10. Available at: https://kclpure.kcl.ac.uk/portal/files/62089077/TeachersPerspectivesIFIPWithAuthorDetailsv2.pdf

[6] Parmit K. Chilana, Celena Alcock, Shruti Dembla, Anson Ho, Ada Hurst, Brett Armstrong, & Philip J. Gu. (2015). Perceptions ofnon-CS majors in intro programming: The rise of the conversational programmer.*CHI Conference on Human Factors in Computing Systems*, *Montreal QC*, Canada. Available at: https://dl.acm.org/purchase.cfm?id=3174085

[7] Tacksoo Im, Sebastien Siva,Jason Freeman, Brian Magerko, Greg Hendler, Shelly Engelman, Morgan Miller, Brandi Villa & Tom McKlin. (2017). Incorporating music into an introductory college level programming course for non-majors.*IEEE Integrated STEM Education Conference, Princeton*, NJ, USA.Available at: https://ieeexplore.ieee.org/document/7910246/

[8] Parmit K. Chilana, Rishabh Singh,& Philip J. Guo (2016). Understanding conversational programmers: A Perspective from the Software Industry. *Proceedings of the 34th Conference on Human-Computer Interaction*.1-11. Available at: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/06/chi16-3.pdf

[9]Azad Ali & David Smith. (2014). Teaching an introductory programming language in a general education course. *Journal of Information Technology Education: Innovations in Practice*, *13*, 57-67. Available at: http://www.jite.org/documents/Vol13/JITEv13IIPp057-067Ali0496.pdf

[10]Dennis R. Goldenson. (1996). Why teach computer programming?Some evidence about generalization and transfer. *National Educational Computing Conference*, Boston. 1-16. Available at: ftp://ftp.cert.org/pub/emg/transferPaper/necc'96.pdf

Copyright © 2018. IJEMR. All Rights Reserved.