A Collaborative approach to scaffold programming efficiency using Spoken Tutorials and its evaluation

[a]Kiran L.N. Eranki, [b]Kannan M. Moudgalya
IDP Educational Technology
Indian Institute of Technology Bombay
Powai, Mumbai 400076, India.
Email: [a]erankikiran@iitb.ac.in, [b]kannan@iitb.ac.in

***Abstract*—Computer scientists and educators have argued that teaching programming skills helps enhance thinking skills and good problem solving aptitude. Learning to program is difficult for many students. Although several factors that affect learning have been identified over the years, we are still far from a full understanding of why some students learn to program easily and quickly while others flounder. This paper addresses this challenge using spoken-tutorials as a collaborative scaffolding tool to develop programming efficacy and evaluation of the students programming abilities. The results of the study show that programming self-efficacy is influenced by prior experience, and the students' mental models of programming influences their self-efficacy, and both the mental model and self-efficacy have a direct effect on overall perfomance of the students. Mastery of programming skills in one these languages also contributed in improved performances in other languages.**

***Keywords*—Programming ability, spoken-tutorials, collaborative learning, high-order thinking, constructivist approach**

I. Introduction

The evidence of dropout and failure rates in introductory programming courses at the university level is due to the fact that learning to program is a difficult task. Decisions about majoring in computer science and related fields are often determined by a students success or failure in the introductory course. According to [1] even after attending the introductory programming classes many students fail to program at the end of the course. And it was claimed that programming skill acquisition happens from basic constructs to writing [2] and building logical flow is an important precursor to acquire writing. It is also noticed that the result varies across universities. And in order to assess the achievement of students, we think that skills such as modifying an existing code or writing a program from scratch are missing in the evaluation process [3].According to [4], self-efficacy is considered as judgmental decisions of ones capabilities to organize and execute courses of action required to attain designated types of performance. In learning situations, self-efficacy influences the use of cognitive strategies while solving problems, the amount of effort needed, strategy adaptation, the levels of perseverance in the face of failure, and the ultimate performance outcomes [4]. According to self-efficacy theory [5], [4], judgments of self-efficacy are based on the individuals performance achievements, observing the performance of others, verbal persuasion, and physiological reactions towards others to judge their own capableness and vulnerabilities.They are critical in debugging a process when things go wrong because the mental models support the person in reasoning about and localizing possible faults [6].In this paper, we report the result obtained by assessing conceptual understanding of basic programming concepts and propose a method to evaluate self-efficacy based on the scores obtained using spoken tutorials. These tutorials have previously been shown to be an effective instructional material [7], [8], [9].Although several studies have been conducted on comprehension of programming skills, little work (if any) has been applied to discover the individual traits of a particular language that cause the most difficulty for novices. Studies into whether the choice of a programming language affects program apprehension is still not clear. It has also been found that different notations facilitate the understanding of different kinds of information found in programs [10]. Other studies [11] have conducted research into the types of mental models formed by both novice and expert programmers and how such models affect their understanding of the problem and its solution. This paper investigates an influence of programming concepts on programming skills and their correlations using Scilab [12], Python [13] and PHP [14] as programming subjects of analysis. Cognitive and social cognitive theory[15] proposes mental models and self-efficacy as two important constructs to teach programming. This study also attempts to address the influence of self-efficacy and mental models on novice programmers, explore the relationship between these concepts, and investigate the effect on course performance.

Spoken Tutorials

A spoken tutorial [16] is a screencast that captures an expert demonstrating an activity along with narration. A ten minute spoken tutorial can have more than about one hundred screen transitions. As a result, the screencast is the most effective way to create such an instructional material, compared to all other methods, for example, demonstrating how to write a program in Scilab to generate a matrix or a graph. Besides, these tutorials can also be localized to reach large masses with a very small effort, if the original is properly created [7]. This makes spoken tutorials accessible even to people who are weak in English. The use of English screen shots ensures that these learners do not lose the employability. The recently launched national mission on education through ICT [17] promotes such

COLLABORATECOM 2012, October 14-17, Pittsburgh, United States

DOI 10.4108/icst.collaboratecom.2012.250466

initiatives.We try to analyze the efficacy of spoken tutorials to teach programing courses

II. Research Methodology

Research Questions: Spoken tutorials have been chosen as the candidate to understand the programming efficacy of the learner. The research questions examined in this study are:

- Does programming by example or demonstration contribute to programming efficacy and high order thinking skills of students?
- Does programming self-efficacy contribute to programming competency?

Sample: The sample for the study comprised 300 non-computer science students from three different engineering colleges belonging to three different universities with different non-computer-science disciplines. All of them had a basic knowledge of computing and Internet skills. Sample comprised of 252 male and 48 female participants.These students underwent workshops to learn Scilab, PHP/MySQL and Python. Each workshop comprised a two hour session with pre and post workshop assessments and individual assignments for each tutorial.The tutorials covered basic programming concepts such as variable assignments, functions, constructs (looping and non-looping), syntax and semantics of the languages.

Self Efficacy Questionnaire: The participants of the workshops answered a computer programming self-efficacy questionnaire [18]. To judge their abilities on programming skills and the mental makeup to accomplish them, they were asked 32 questions on the following four topics: simple programming abilities, complex programming abilities, perseverance and self-regulation. The responses were recorded on a seven point likert scale ranging from 1=not confident at all to 7=absolutely confident. The same questionnaire was administered before the workshop and after the workshop. Through this mechanism, it is possible to assess the efficacy of the workshops, as perceived by the students. Only after completing the questionnaire, could the students participate in pre and post workshop assessment tests. Only the students who did well in the post workshop tests were eligible to receive a certificate.

Assessment Tests: All participants were evaluated through pre and post workshop assessment tests. The post workshop test comprised 30 questions, while the pre-workshop test consisted of ten questions. All were randomly selected from the same data bank of questions. The performance in the post workshop tests gives rise to most of the data presented in this article.Randomly selected questions were presented to five experts for their feedback on the reliability of the tests. The feedback suggests that the questions have a high inter-rater reliability(Cronbach $\alpha = 0.98$). Basic programming, Sequence, Prediction1, Prediction2, Explain, Generate1 and Generate2 were the question types used in the analysis. Multiple choice type questions were posted in the assessment tests. We will now present a few samples of these questions.

a) Predict1: The participants were asked to predict the output of the given code to test their grasp of looping constructs:

```
------------------------
Predict the output:
s=5;
while s>3
disp(s^3);
s=s-3;
end
----------------------
Answer:s=125
--------------------
```

As the answer is a number that the students have to arrive at, random guessing is completely eliminated in this type of questions.

b) PR2.Sequence: The participants were asked to identify the missing code by replacing A and B, with correct code options to count the number of words in a string.

```
----------------------
<?php
$string = ereg_replace("  +"," ",$string);
A
$numwords = explode(" ", $string);
B
?>
Options for [X]:
1. $string = strip($string);
   $numwords = lower($numwords);
2. $string = round($string);
   $numwords = upper($numwords);
3. $string = trim($string);
   $numwords = count($numwords);
4. None of them;
----------------------
Correct option: 3
----------------------
```

c) PR4.Modification1: Spoken tutorials teach programming through examples or demonstration. In addition, as a part of the assignment, the students are provided a sample code to effect suitable modifications. Both of these can be assumed to be effective if the leaner does well in assessment tests that check the programming capability. One way to check the programming capability is to evaluate whether the students can identify the correct program code from given codes. The difference between Modification1 and Modification2 is that Modification1 [MWL] is loop-free while Modification2 [ML] involves loops. A sample question on Modification1 is shown below, where students were asked to identify the correct program code:

```
------------------
Karan want to transfer value from i to j
without using a third variable.Select the
correct code option.

<?php
var $i, $j;$i=10; echo "The value of i: $i";
echo "Value of i, j is:", $i," ",$j;
?>
```

```
A.$i+=$j; $j=$i-$j; $j*=$j;
B.$i+=$j; $j=$i-$j; $i--;
C.$j=$i-$j; $i++;
D.$i+=$j; $j=$i-$j; $i+=$j;
-------------------------
Correct Answer: D
-----------------------------
```

Unless the students selects the correct program code, their answer is marked wrong.

d) PR6. Generat1: Generate1 [GWL] and Generate2 [GL] evaluate the capability of the students in writing correct programs, with and without loops, respectively. A sample generate1 is given below:

```
-----------------------------------
Write a program to sum the digits
of a five digit number.
(Hint: Use the modulus operator %).
-----------------------------------
```

This is a multiple choice question with each choice containing a different code. The students have to select the correct option. We are working on improving the types of questions so as to reduce the dependence on random marking.

Self-efficacy Analysis: The score obtained in the questionnaire is considered to be the self efficacy score (SES) of a participant of this study. The 600 SES scores of the 300 participants in both pre and post questionnaire were analyzed. The results of analysis of varience (ANOVA) was significant, $F(3,296) = 14.96$, $p<.0001$. This may be because of the following two possibilities: (1) the participants are heterogeneous (2) there is a significant difference in the pre and post SES scores.The 300 students who participated in this study were randomly put into four different groups, called, A, B, C and D. We have found that there is a significant difference between group B and the other groups. Tukey range test confirmed this observation.The results indicate that the group with the highest PreSE (B, group mean = 141.35) experienced the least increase in PostSE. This group differed significantly ($p<.05$) from groups A, C and D, which experienced much larger increases in efficacy. Groups A, C and D did not differ significantly from one another. The members of group B, with the highest PreSE, claimed that they had exposure to some other programming languages previously, for example, through their schools or through some workshops. The other groups showed larger improvements in SES, with the values for the groups A and D being much larger. This shows that the novice learners improved much more significantly, compared to experienced learners. The improvements are in perception, comfort to handle programming situations such as writing, modification and sequencing.As the SES improved for everyone and for about half the population significantly, we believe that the spoken tutorial based learning of programming is effective.

Collaborative approach to scaffolding: Evaluation of student programming skills requires a lot of questions to assess their competency. Covering all FOSS systems and also the various programming topics is a Herculean task. The only way to achieve any meaningful progress in this effort is to resort to collaborative question bank generation and review. Where students can contribute as well as review the questions and this approach also scaffolds the programming skill of the learner by engaging them in continous learning process. This also helps to support peer-reviews and collaborative learning. Though we had 85% and above performance in post-tests as against the 46% performance in pre-tests. We have found that students who scored above 80% in post-tests contributed more number of questions and showed interest in review and rating of the questions with reasons to support their rating.

III. Influence of programming concepts on skill acquisition

We further discuss the influence of programming concepts on skill acquisition of an individual by performaing a comparative correlation study of each and every concept selected for the study and its impact on the corresponding programming skill.

Skill Analysis

We observe that skill related to writing functions [WF] is correlated with Generate1 (0.42) skills. Writing is simple and has some similarity with Generate1. Modification1 is correlated with Generate1 (0.41) and Identify (0.58) skills. Modification2 is correlated with Generate2 (0.55) and Explain (0.51) skills. Generate1 is correlated with Modification2 (0.72) and Explain (0.62), Modification2 is correlated to Explain (0.57), and Generate1 (0.33). Recall that in the Modification questions, the student has to interpret the logical flow of the code to predict the output. As it is correlated to the code writing (Generate), the student has to visualize the code to the answer these questions. On the other hand, these four skills are only weakly correlated with Predict1 (0.06 to 0.16) and Predict2 (0.07 to 0.39). Possible interpretation of this is that for solving Modification and Generate, students need some construction ability and Predict is less useful for it.In the pre-test, the correlations are as follows. Writing is correlated with Generate1 (0.26), Programming2 (0.36). Modification1 is correlated with Generate1 (0.46), Modification2 (0.36), and Generate2 (0.33). Modification2 is correlated with Modification1 (0.36), Generate1 (0.46). Generate1 is correlated with Modification1 (0.46), Modification2 (0.33). Generate2 is correlated with Identify[IDE] (0.36). These four skills are less correlated with Predict1 (0.07 to 0.33). Correlation patterns in the pre-test are less visible than those in the post-test. Possible interpretation of this is that in the pre-test students were not familiar with the question types and skills were not adequately developed. So the students solved the questions from mainly clues. In the post-test the skills of the students were developed and the proper skills were used to solve the questions.

Comparative Study of Concepts

Descriptive statistics using correlations above 0.7 ($p=0.05$) among the concepts and concepts to skills, one can make

the following predictions: Constructs(Looping)(C7) has a significant correlation effect on the Assignment(C5),and Functions(C8). The understanding of the nested loop can enhance the understanding of C8 and C5. Similarly, Logical Flow(C4) has a significant correlation with Functions(C8), Call by value/reference(C1). because understanding the logical flow of data also helps in writing better functions and use of call by value/references(C1). Significant correlation effect is also shown in between Data type validation(C3) concept and Syntax and Semantics(C10). because the data type of the variables defined while writing a program also vary according to the syntax and semantics of a language. So a better understanding of C10 helps in implementing the correct C3 concept. Similarly,understanding of the logical expression (&, ||, and !) may be considered supplemental, but the understanding is important for understanding C10 and C3, C8, C7 (maybe, conditional part of it) concepts.Students taught with PHP concepts on functions, constructs (Loop/Non-Loop) would actually contribute to better understanding of similiar concepts in Scilab and Python. The converse, however, is not necessarily true, as Scilab is not really a programming language. Moreover, Scilab closely resembles the popular Matlab.Students who have gained proficiency in basic programming skills such as assignments, constructs, functions, sytax and semantics of any one of these languages would actually be performing well in any of the other languages, because the conceptual idea of implementation is the same though it varies in the keywords, syntax and semantics of the languages.

IV. Conclusions

Using the data collected from students who underwent spoken tutorial based workshops on Scilab, PHP and Python, we studied self efficacy, and skill and concept analysis.Students who have prior experience in programming seem to show high self efficacy. Novices have found the spoken tutorials to be more useful than those who had prior experience. Pedagogical aspects of self efficacy have also been studied.A statistical approach has been used to discover relationships between programming skills and programming concepts taught to students using spoken tutorials. Results of the descriptive analysis shows that programming skills such as Modify1, Modify2, Programming1, and Programming2 are very different kind of skills from sequencing. Some of these observations are in line with the earlier findings [2]. The skills also get matured with the development of abilities of students.Spoken tutorial based approach of programming by examples or demonstration helps students practise several times till they get a mastery of the underlying concepts. For a novice, programming by demonstration provides a constructivist approach to understand concepts and to encourage self-learning. This is because the spoken tutorials do not restrict the learners by time, location and resources. More work is required to form concrete conclusions relating to programing skills and concepts and interplay between different programming languages.

Acknowledgments

This work was partly funded by the National Mission on Education through ICT, MHRD, Government of India, through the Talk to a Teacher project. We thank the project staff members of spoken tutorial project and the participants of the study for their time and efforts.

References

[1] R. McCartney. A multi-national study of reading and tracing skills in novice programmers. In *Working group reports from ITiCSE on Innovation and technology in computer science education, ser.*, pages 119–150. ITiCSE -WGR'04, ACM, March 2004.

[2] P. Robbins J. Whalley. Relationships between reading, tracing and writing skills in introductory programming. In *Proceedings of the Fourth international Workshop on Computing Education Research, ser*, pages 101–112. ICER'08, ACM, March 2008.

[3] E. Thompson and R. Lister. Code classification as a learning and assessment exercise for novice programmers. In *Proceedings of the 19th Annual Conference of the National Advisory Committee on Computing Qualifications*, pages 291–298. NACCQ'06, ACM, July 7-10 2006.

[4] A. Bandura. *Social Foundations of Thought and Action*. Prentice Hall, Englewood Cliffs, NJ, 1986.

[5] A. Bandura. Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review*, 84(2):191–215, April 1977.

[6] Robert Allen and David Garlan. A formal basis for architectural connection. *ACM Trans. Softw. Eng. Methodol.*, 6(3):213–249, July 1997.

[7] K. M. Moudgalya. Spoken Tutorial: A Collaborative and Scalable Education Technology. *CSI Communications*, 35(6):10–12, September 2011. Available at http: //spoken-tutorial.org/CSI.pdf.

[8] K. L. N. Eranki and K. M. Moudgalya. Evaluation of student perceptions and interests using spoken tutorials in online courses. In *International Conference on Advanced Learning Technologies*, ICALT 2012, Rome, Italy, 4-6 July, 2012. IEEE.

[9] K. L. N. Eranki and K. M. Moudgalya. Evaluation of web based behavioral interventions using spoken tutorials. In *Technology for Education*, T4E 2012, Hyderabad, India, 18-20 July, 2012. IEEE.

[10] S. Wiedenbeck. Novice comprehension of small programs written in the procedural style. *International Journal Human-Computer Studies*, 51(1):71–87, June 1999.

[11] J.M. Burkhardt. Mental representations constructed by experts and novices in object-oriented program comprehension. In *Human Computer Interaction*, pages 339–346. INTERACT'97, July 1997.

[12] SCILAB. www.scilab.org. seen on 20 Sept 2012.

[13] Python. www.python.org. seen on 20 Sept 2012.

[14] PHP. www.php.net. seen on 20 Sept 2012.

[15] J.J. Canas and P Gonzalvo. Mental models and computer programming. *International Journal of Human-Computer Studies*, 40(5):795–811, June 1994.

[16] Spoken Tutorials. www.spoken-tutorial.org/new. seen on 21 May 2012.

[17] NMEICT. http://www.sakshat.ac.in/. seen on 20 Sept 2012.

[18] Wiedenbeck S. Ramalingam V. Development and validation of scores on a computer programming self-efficacy scale and group analyses of novice programmer self-efficacy. *Journal of Educational Computing Research*, 19(4):365–379, June 1998.