An Integrated Approach to build Programming Competencies through Spoken Tutorial Workshops

[a]Kiran L. N. Eranki, [b]Kannan M. Moudgalya
IDP Educational Technology,
IIT Bombay, Powai, Mumbai 400 076, India.
email: [a]erankikiran@iitb.ac.in, [b]kannan@iitb.ac.in

***Abstract*—This work investigate the implications of using visualization tools along with spoken tutorials to build comprehension and debugging skills of novice programmers. And also, evaluates the self-learning approach to teach programming skills to the students. Qualitative and Quantitative studies were conducted using spoken tutorial workshops on Java and C++ courses. Results of the study have shown improvement in programming skills and conceptual understanding when subjected to program visualization along with spoken tutorials.**

***Keywords*—spoken-tutorial, programming, visualization, comprehension, debugging**

I. Introduction

Programming competencies mainly include programing comprehension and debugging skills which are the foundations of programming skill. These skills are difficult both for novice programmers to learn and computer science educators to teach[1]. A study conducted by ITiCSE group to assess the programming ability of first year engineering students found that average score was only 20%[2] which also lead to high dropout rates, of around 30-60% associated with CS courses. Considering the growing demands of IT workforce for skilled programmers, programming education has a significant role to meet these requirements. Inadequate problem-solving abilities and lack of essential mental models of key programming concepts lead to misconceptions and difficulties in solving programming problems[3]. Researchers and instructors emphasize the development of conceptual mental models of various key programming concepts. Most students carry a pre-defined set of ideas on computing concepts such as object assignment, where students think assigning a value to variable A and transfer the value of variable A to variable B makes variable A lose its actual value. And several such misconceptions also exist in object reference concepts as well. Constructivist theory argues that traditional teaching approaches are passive and do not provide adequate opportunities for students to recognize the errors and resolve the misconceptions of the concept[1]. Visualization techniques provide opportunities to address misconceptions on object assignment and reference concepts discussed earlier. In this pursuit, we examine the implications of visualization tools used along with spoken tutorials to examine programming competencies of the learner. The paper is structured as follows: Section 2 presents the relevance of spoken tutorial workshops in programming education. Section 3 describes the research methodology used to conduct the study, followed by the results and conclusions of the study are presented.

II. Spoken Tutorial Workshops

In this section, we will give a brief overview of various free open source software(FOSS)courses conducted through Spoken tutorial based Education and Learning on FOSS(SELF) workshops[4]. We also flag various programming difficulties expressed by students while learning these courses through workshops.

A. Self-Learning Tutorials

A spoken tutorial is a screencast with a running commentary of an expert demonstrating how to write a program or execution method. Novice learners can easily replicate the tutorial exercises and be able to program. A ten minute spoken tutorial can include more than one hundred screen transitions. As a result, these video tutorials qualify as comprehensive instructional material, compared to other methods[4]. These tutorials provide a self-learning approach to learn a new concept by practicing while watching a tutorial. Hence, these workshops can be conducted by non-expert organizers as well. About 3,000 workshops have been conducted so far since June 2011. At present, about 300 workshops are conducted every month. These workshops are at present conducted on the following FOSS systems: Linux, Python, Scilab, LaTeX, C/C++, Java, LibreOffice and PHP-MySQL.

B. Programming difficulties due to Self-Learning Approach

SELF workshops conducted to train students on programming were assessed at recall, understand and apply levels of blooms taxonomy. During this study, the participants were provided with a collection of short C program codes, such as generate Fibonacci series of numbers. These programs were at four distinct levels of complexity: 1) Basic while-loop construct. 2) Iterative looping. 3) Recursive procedures. 4) Embedded recursion procedures were used. Students were asked to fill the missing code. Most of them solved first two levels of non-recursive complexity. but had difficulty solving the last two levels due to recursive procedures. As programming concepts are taught using self-learning mode, students had a difficulty in understanding programming concepts above apply level such as recursion, inheritance. Students expressed concerns to solve the programming tasks[5]. Some participants

tried to understand individual lines of code, and ignored the context built by all lines of code that was previously executed. And some participants used their prior knowledge or experience to understand programming concepts. In the remaining sections of the paper we discuss how we addressed the programming difficulties of the students and improved their programming competency through these workshops.

III. Research Questions

This study investigates the effectiveness of visualization tools in improving programming competency. Students have programming difficulties while studying through self-learning workshops. So visualization tools were added along with spoken tutorials to address these difficulties. These tools visualize the program execution in a step-by-step manner through animation allowing the learner to practice and improve their programming skill. The research questions examined in this study are:

1) Does program visualization contribute to programming competency?
2) Does visualization help learner build a reasonable mental model of the program?

Sample and Process

The sample comprises 160 students from non-computing discipline of a local engineering college. A randomized assignment of sample was done based on their order of registration and discipline of study. All of them had a basic computer literacy. The students were distributed into two groups[A,B] of 40 students each for Java and two groups of 40 students each for C++ course conducted through Spoken tutorial workshops. These workshops are of two hour duration with pre and post workshop assessments and individual assignments for each tutorial. Five concepts were selected for both Java and C/C++ tutorials for this study. The tutorials covered programming concepts such as variable assignments, recursions, polymorphism, pointers and inheritance topics. All students watched spoken tutorials in both the workshops. The experimental Group(A) students solved object assignment and reference exercises using visualization tool along with tutorials. They also received concept-wise diagnostic feedback while solving the exercises using the tool. Whereas, control Group(B) students solved the same exercises without using visualization tool and No diagnostic feedback was provided. After the workshop, both the groups attended post-test and submitted workshop feedback questionnaires.

IV. Role of Visualization in Programming Competency

Visualization Tool

The current study investigates the effectiveness of program visualization in improving programming skills. Visual Programming allows the user to specify a program using graphics and animated tool options, unlike conventional programming that consist of a text-based editor with no graphical outputs[6]. Visualization tools help novice learners understand logical flow of data within the program. Codeasy, open-source java application built on jeliot 3.1[7] engine was used in the workshop. This tool consists of four sections namely- Methods, constants, expression instance/array sections which help the learner to visualize the data flow transitions. Snapshot of pop-up quiz feature in visualization tool is shown in Fig. 1. We considered jeliot engine for teaching java, as it supports both animation and object-oriented programming. We customized it to suit our workshops by adding features such as pop-up quiz questions and dynamic addition of compiler headers to coding section. Diagnostic feedbacks are provided at each and every step of the program execution while assessing their programing concepts.

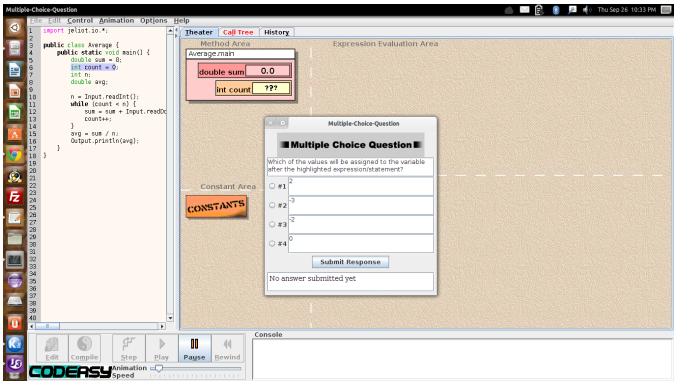


Fig. 1. Codeeasy: Pop-up Quiz feature snapshot

Programming Competency Questionnaire

We examined the programming skill of the learner through programming competency questionnaire. This questionnaire had both close and open ended questions which require a descriptive explanation of program code provided. Participants programming concepts were assessed based on data representation and verbal data gathered through interviews that describe their thoughts of program execution. Students were asked to solve the object assignment and reference exercises, which evaluate manipulation of object assignment and references using a method.

Analysis of Competency Questionnaire

The effect of visualization tool on programming competency has been analyzed using the Analysis of Variance (ANOVA). The effect of visualization tool was statistically significant ($F(3,80)=4.68$, $p<0.05$) on improving programming concepts. These results also explain novice programmers transition from misconceptions to reasonable understanding of the concepts. However, no significant gender difference was noticed on programming competency among the groups. Group(A) students showed significant improvement in comprehension and debugging skills (Java:68%,27 and C++:83%,33) as shown in Table I. While no significant improvement was noticed in Group(B) for both the courses. The participants in both the groups showed keen interest in visualization tools but only Group(A) was provided with visualization. The effect of prior

experience in programming also showed effect($F(3,80)=8.46$, $p>0.001$) on improved performance in 3 participants of Group(B), who attended a programming course either in school or summer camps. Considering the novices conceptual understanding needs to be build through formal training or teaching aid to learn new concepts. So visualization tools were very useful to them as observed. And this also confirms the effect of visualization on improving programming skills.

TABLE I
IMPLICATIONS OF VISUALIZATION ON PROGRAMMING COMPETENCY

Number of Students in each Group					
Course	Group	Compreh Improvmnt	Debug Improvmnt	No-Improvmnt	N
Java	A	12	15	13	40
	B	5	9	27	40
C/C++	A	20	13	7	40
	B	12	9	19	40

V. ROLE OF VISUALIZATION IN MENTAL MODEL

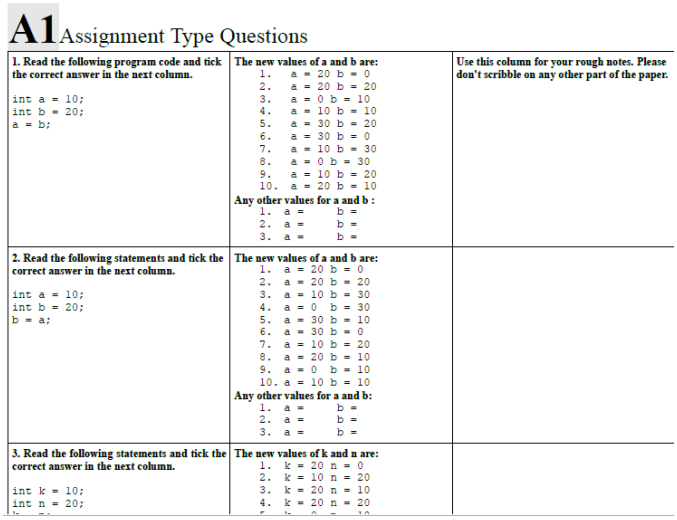
In this section, We present the effects of visualization in improving the student mental model of the program and its evaluation through mental model questionnaire.

Mental Model

We define the student understanding on programming concepts as student mental models. Compared to mental models of physical devices, mental models on programming concepts are more difficult[8]. While physical devices are visible and tangible, it is relatively easy for users to create a mental model of how the devices operate. On the other hand, programming concepts are invisible and difficult to image[6]. Students cannot see what is happening in the computer when a program is executed[1]. As a result, students hold several misconceptions on program execution[3]. Program visualization provides a potential solution to address this problem. We examined the misconceptions held by the learner using mental model questionnaire.

Mental Model Questionnaire

Earlier studies on programming competency using spoken tutorials have shown 50-65% performance on comprehension and debugging tasks[5]. All students attempted mental model questionnaire followed by pre-workshop test before participating in the workshop. This questionnaire has been validated by several researchers earlier[8] in CS education research. This questionnaire mainly has two types of questions-A1.assignment and R1.references. Snapshot of Mental Model Assignment Questionnaire has been shown in Fig. 2. After two hour workshop, students were asked to fill the questionnaire once again. While workshops without visualization tools lacked this opportunity. As a result, student had a difficulty to comprehend higher levels of programming competency through assignment and reference exercises. The performance of Group(A) students in post-test also showed significant improvement both in Java(SD=86.35, 40) and C++(SD=76.84, 40) workshops as shown in Table II.


A1 Assignment Type Questions

1. Read the following program code and tick the correct answer in the next column.

```
int a = 10;
int b = 20;
a = b;
```

The new values of a and b are:
1. a = 20 b = 0
2. a = 20 b = 20
3. a = 0 b = 10
4. a = 10 b = 10
5. a = 30 b = 20
6. a = 30 b = 0
7. a = 10 b = 30
8. a = 0 b = 30
9. a = 10 b = 20
10. a = 20 b = 10

Any other values for a and b :
1. a = b =
2. a = b =
3. a = b =

Use this column for your rough notes. Please don't scribble on any other part of the paper.

2. Read the following statements and tick the correct answer in the next column.

```
int a = 10;
int b = 20;
b = a;
```

The new values of a and b are:
1. a = 20 b = 0
2. a = 20 b = 20
3. a = 10 b = 30
4. a = 0 b = 30
5. a = 30 b = 10
6. a = 30 b = 0
7. a = 10 b = 20
8. a = 20 b = 10
9. a = 0 b = 10
10. a = 10 b = 10

Any other values for a and b:
1. a = b =
2. a = b =
3. a = b =

3. Read the following statements and tick the correct answer in the next column.

```
int k = 10;
int n = 20;
```

The new values of k and n are:
1. k = 20 n = 0
2. k = 10 n = 20
3. k = 20 n = 10
4. k = 20 n = 20

Fig. 2. Snapshot of Mental Model Assignment Questionnaire

TABLE II
VISUAL(VT+ST) VS NOVISUAL SPOKEN TUTORIAL WORKSHOPS

FOSS	Group	PreTest	PostTest	Mean	SD	N
C/C++	A(VT+ST)	62.15	186.25	122.10	86.35	40
	B(ST)	64.35	141.25	78.90	55.50	40
Java	A(VT+ST)	62.33	172.55	109.22	76.84	40
	B(ST)	63.25	124.55	64.30	45.50	40

Analysis of Mental Model Questionnaire

Depending on the performance of the learner on mental model questionnaire and post-test the effectiveness of the visualization is being analyzed as shown in Table II. We found that most of the students from Group(A) showed better understanding of programming concepts and performed significantly better (N=40, 82%) on assignment,reference exercises and post-test programming tasks than those of Group(B) students. Few students from Group(B) performed well on more advanced concepts of reference assignment without any practice or exposure to visualization tool. This could have been due to prior experience of the learner with the language. We have found Group(A) participants performance better then Group(B) both for Java and C/C++ courses. The test questionnaires cover three close-ended questions and one open-ended question. The close-ended questions tested the participants on execution result of a program with multiple reference assignments(topic: Inheritance). A sample open-ended questions has been shown below.

```
\\open-ended question\\
A Student class defined with 'name' as
string object. Constructor 'new Student("ajay")'
is used to create an object of Student class.
'changeName("kartik")' method is used to change
the name to "kartik".

Describe what happens when following statements
are executed. You may use both text and diagrams
in your answer.
Student a;
```

```
Student b;
a = new Student (ajay);
b = new Student (pavan);
a = b;
b.changeName(kartik);
a.name = _____; b.name = _____;
```

Concept-wise performance among the groups also showed higher Group(A) performance in inheritance(N=40, 84%), polymorphism(N=40,80%) and Arrays (N=40, 83%) as compared to Group(B) for the same concepts(N=40, 67%; N=40, 60%; N=40, 68%) as shown in Fig. 3.

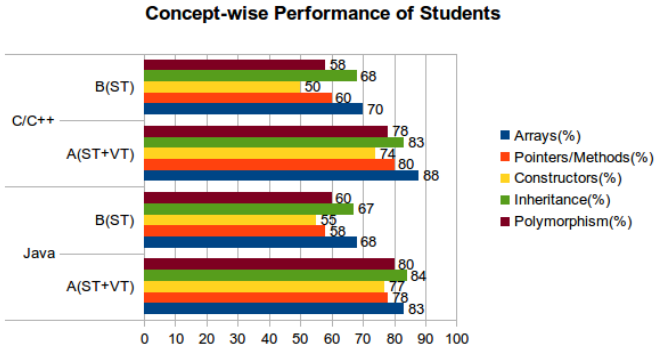


Fig. 3. Concept-wise performance of students

Workshop Feedback Questionnaire

Workshop Feedback questionnaire was also administered at the end of the workshop. Feedback scores also showed an acceptable usability score of p=74.1% for non-visualization groups and p=86.6% for the visualization group. Based on the usability study and learner feedbacks, we have found both the methods of teaching programming courses seem to be acceptable to the users of spoken tutorials. As mentioned earlier, workshops with visualization tools had interactive feedbacks and graphical representation of program execution. As a result, student misconceptions generated during self-learning workshops have an opportunity to correct and improve their programming skills.

VI. Conclusions

The study presented in this paper addressed the student programming difficulties caused due to self-learning workshops on Java, C++ concepts. And also showed the application of visualization tools to improve programming competencies. This has also contributed to evaluation of programming courses and learners feedbacks for improving workshops. We hope this study will contribute to better understanding of the needs of programming education. However, the results also show that some students did not show any change in their understanding on reference type questions after using the proposed materials while most of them showed a change. A possible reason would be, some students with limited comprehend skills might need more time to build programming concepts. Despite having experienced program visualization, their current understanding is not sufficient enough to solve the tasks. Extensive learning and practice in programming might help them improve their skills to solve the tasks and construct a viable understanding of programming concepts. This study also needs to be extended further to other programming languages and visualization tools to generalize our claim on development of conceptual understanding and programming competency through program visualization.

Acknowledgement

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 this study for their time and efforts.

References

[1] M.. Ben-Ari. Constructivism in computer science education. 20(1):45–73, 2001.

[2] Almstrum V. McCracken, M. A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. 33(3):125–140, 23-25 April 2001.

[3] P. Bayman and R. E. Mayer. A diagnosis of beginning programmers' misconceptions of basic programming statements. *Communications ACM*, 26(9):677–679, 1983.

[4] 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.

[5] K. L. N. Eranki and K. M. Moudgalya. A collaborative approach to scaffold programming efficacy through spoken tutorials. In *Intl.Conference on CollaborateCom 2012*, Pittsburgh, PA, USA, 23-26 October, 2012. IEEE.

[6] B.A. Myers. Taxonomies of visual programming and program visualization. *Journal of Visual Languages and Computing*, 1(4):97–123, 1999.

[7] Gomes A Marcelino, M. Using a computer-based interactive system for the development of basic algorithmic and programming skills. In *International Conference on Computer Systems and Technologies (CompSysTech2004)*, CompSysTechT 2013, Rosse,Bulgaria, 17-18 June, 2004. IEEE.

[8] Bornat R. Adams R Dehnadi, S. Meta-analysis of the effect of consistency on sucess in early learning of programming. In *21st Annual Psychology of Programming Interest Group Conference*, Limerick,Ireland, 2009. IEEE.