Analysis of Teaching Computer Programming in Indian Context

P. Chenna Reddy
Professor of Computer Science and Engineering
JNTUA College of Engineering, Pulivendula
YSR District, A.P., India
E-mail: pcreddy1@rediffmail.com

Abstract— **Engineering education is given significant importance in India. A fundamental component of a computer science and engineering curriculum is computer programming and is generally studied at first year under-graduation. It is not only a difficult course but is the basic course which can affect the student prospects in the entire engineering program. It places the student in a groove and enables him to move along the correct path. Unfortunately engineering education in India is going through a turbulent period and there is need to correct it. Programming is a subject which has become essential across the disciplines. This paper analyses the reasons for the difficulties experienced by first year programming students, reviews current methods of teaching programming, identifies effective ways of teaching programming, and provides guidelines to fine tune teaching programming to individual requirements. It assumes Indian environment.**

Keywords— Engineering; Programming; Under-graduation; Teaching; Syllabus

I. Introduction

Globalization is aimed at reducing the gap between the nations and enables people and markets to view the world from the same perspective. Though the main aim may be considered as enabling the international trade, there are many other goals which are also important. Universal recognition of the degrees offered by the different nations is one of the main results of globalization. Different nations have different standards of education making the mobility of the students and intellectuals difficult. Washington accord as in [1] is a global agreement among bodies responsible for accreditation of engineering programs in various nations. Computer Science and Engineering is one programme which is studied by students in different countries.

Computer science and engineering by being a professional engineering field is also influencing all other engineering and non-engineering fields. Computer programming is an integral part of a computer science curriculum. Since programming is essential requirement for the rest of the computer science program, it is studied at first year under-graduation level. Moreover, computer programming is essential for many other non-engineering fields also. Many students find programming difficult to understand. Failing to perform better in computer programming subject discourages the students and affects their performance in the rest of the subjects in computer science curriculum. Many teachers also feel it difficult to teach programming to students because the dynamics of programming subject are different from other traditional subjects.

Various programming languages are tried as introductory programming courses as in [2] [3]. Attempts are made by varying the type of programming languages like procedure oriented, object oriented and functional programming languages. C, C++, and Java are tried as introductory programming languages. The content of the syllabus is also changed quite often to make learning and teaching process easy. Various teaching methods are also tried like teaching using black board, teaching using overhead projectors, LCD projector, teaching in a laboratory with each student having his own computer, teaching using some graphical tools etc. Different Evaluation systems are also tried to find the student limitations and motivate the student to improve the interest in the subject. None of the methods resulted in much success and further research is on to effectively teach programming.

There is no universal agreement about the type of the programming language and also particular programming language which is suitable for first year under-graduate students. The syllabus is revised regularly in many universities. But, unfortunately, the content of the programming subject at first year level is rarely revised. No thought is made about the subjects at first and second year levels. They are considered as core subjects whose syllabus need not be modified. Main concentration is on subjects at third and final year level because they are considered as advanced subjects which keep track of the advancements in the concerned field of engineering.

The black board is the traditional medium for teaching. The use of projectors is viewed as an alternative to black board and is used to present the content in the form of slides instead of writing on the black board. No attempt is made to use projectors effectively. No attempt is made to teach programming to match with the thought process of the student. The way laboratories are handled is more painful. Programming laboratories have become typing institutes. The list of experiments prescribed for the student is rarely changed. Student just copies the code from the senior's records or from the Internet. Student rarely attempts to write programs on their own and improve programming logic. The net effect of it is student is never permitted to develop his logical abilities, creative abilities, etc.

Age-old methods are used for evaluation because of the practical problems in the implementation of innovative methods. Thanks to the All India Council for Technical Education (AICTE) as in [4] for liberally granting the seats in engineering colleges across the country. Number of students in a single branch of reputed institution is equivalent to total number of students of all branches in an engineering college which is not reputed. Definitely number of staff members and the infrastructure facilities are not improving at that rate leading to deterioration of the standards even in the prominent engineering colleges.

There is no paper, as far as my knowledge goes, which comprehensively covers effectively teaching programming subject to students. Research is limited to advanced subjects. Research on fundamental subjects is often neglected making them difficult to comprehend. This paper considers research in fundamental subjects as essential and hence deals with teaching programming to first year under-graduate students.

This paper doesn't evaluate different programming languages for their suitability to teach to first year under-graduate students. This paper assumes first year under-graduate students unless specified otherwise. Teaching programming to students other than those of under-graduation may have different requirements but this paper doesn't concern about all those requirements. This paper doesn't discuss about why poor performance in programming subject at first year level affects his performance in other subjects.

II. LITERATURE SURVEY

Edsger W. Dijkstra in one of his invited talks as in [5] pronounced that "Computers represent a radical novelty and they should be dealt with in an innovative way". He asked the computer science community to approach programming with a blank mind and linking with our previous experiences is not appropriate. Since programming represents a radical change it is unwelcomed. Computer programming is a gigantic task which is difficult to comprehend by traditional methods. From his perspective, the only thing computers do for us is to manipulate symbols and produce results of such manipulations. He defines program as "the abstract symbol manipulator which can be turned into a concrete one by supplying a computer to it". As a mathematician he views program as a formula, a formula which is larger in size and the programmer task is to derive the formula.

He criticizes the educational policy which is very much relevant. The present educational policies whether designed by Accreditation Board of Engineering and Technology (ABET) as in [6] or National Board of Accreditation (NBA) as in [7] in India involves all the stake holders like students, parents, and industry. Colleges perceive that the purpose of teaching programming is to prepare the students for entry level programming related jobs. He argues that the educational policy has to be influenced by scientific considerations derived from the topics taught and universities should provide for society the intellectual leadership it needs rather than the training it asks for. He severely criticizes the universities for misguiding the students and the way curriculum is designed as infantilization and it is unfortunate because there is no educational progress.

He perceives programming language as a formal system and program execution just checks a model. He goes to the extreme that teaching programming using visualizations should be treated as "contempt of the student body". He argues that this method of teaching does permanent mental damage for the students exposed to it. He recommends the use of imperative (languages in which major effect is achieved through assignment statement) programming languages for the beginners. He discourages use of computers to execute the programs and he recommends treating programming as formal proof that meets the formal functional specifications.

The response of some of the computer science colleagues is equally severe. Dijkstra's view of using formal methods for program verification is impractical. This is particularly difficult when the program size is large. Moreover the mathematical models are not perfect and they fail to mimic many devices precisely. The use of formal proofs has its own limitations and mathematical models for the programs cannot be created by programmer himself because it is a tedious process. In software world, mathematical modeling and testing are considered as complimentary approaches and both are necessary. Though Mathematics and programming are similar, definitely they are disjoint and Dijkstra's view of treating programming as mathematical modeling is not correct. The major problem with giving formal proofs to programs is, in real world it is difficult to precisely say what the user wants particularly when the problem to be solved is complex. When solving a complex problem the method that is generally adopted is building the prototypes of a program and showing to the user by executing it, with the intent of user identifying what he exactly wants. But separate courses can be there on formal proofs. Also use of formal proofs can be considered as one method of teaching with due importance for alternatives. Dijkstra's view of completely understanding how the system should behave is difficult task in the current context of engineering problems.

J. Bruner remarks on the process of education as in [8] are quite interesting. He suggests that "any subject could be taught to any child at any age in some form that is honest". While teaching one should match with learner's capacities which is often not considered. Everybody measures the learning and teaching process using their own yardstick. Mismatch here leads to communication gap which is never addressed. There are two important stakeholders, teacher and student. Teacher centric approach and student centric approach are used to indicate two different approaches.

Bruner as in [9] says curriculum is more for teachers than it is for students though the current trend is towards student-centric approach. A curriculum which cannot move a teacher will have a passive affect on the student. Teacher is the means of communication. If the means is passive then there is no communication. While framing the syllabus Bruner recommends having wide variety of people as members. More important are the psychologists who know the psychology of people of current generation and fill the communication gap between teacher and student which is essential for the success of the education process.

Bruner suggests the division of members involved in framing the syllabus into five work groups: one concerned with "sequence of curriculum", a second with "the apparatus of teaching", a third with "the motivation of learning", a fourth with "the role of intuition in learning and thinking" and a fifth with "cognitive process in learning".

He suggests that knowledge be presented to the student in a structured manner. Unless the details are presented in structured pattern, the details are often forgotten. Basic concepts presented have regenerative character. As humans we forget many things over a period of time. But the basics allow us to regenerate what we have forgotten and also allow the individual to create or derive new knowledge. There is always a link between basic concepts and advanced concepts. Since we already know the basic concepts, learning the advanced concepts is identifying this link.

A teacher should try to establish a link between himself and student. Without establishing a link, success is not guaranteed. A teacher should ask student several questions, particularly not simple or difficult but average questions to understand what the student already knows, what he is expecting from the teacher, and what are his preferred methods of learning. Proper link between student and teacher simplifies the learning process. The learning process of any person should be spiral in nature. He learns the basic concepts, moves from basic concepts to advanced concepts and from advanced concepts to new things in the rest of the life.

Students should be permitted to use intuition. Intuition is defined as immediate cognition. Intuition permits one to arrive at the answer without going through the time consuming derivation process. Knowledge in the concerned field will definitely help in getting the answer directly. Students should be permitted to guess the answers by asking him various questions. He should not be discouraged in making wild guesses because it is part of the learning process. As he gets matured and his self confidence increases, students start making correct guesses. Guessing by trial and error process is slowly converted to guessing based on intuition.

The authors as in [10] present the difficulties faced by first year students in the subject on programming and suggests mechanisms to improve the performance of student in the subject. The programming language chosen for first year is object oriented programming language. The use of different text books, use of different teaching techniques and use of electronic assignments have yielded unsatisfactory results. Lack of understanding of the computer model and lack of previous programming knowledge is one of the reasons for the poor performance of the student. The use of abstract types or terms which are generally not used in natural language is another reason for poor performance. The performance of the student at first year programming subject will affect the overall performance of the student and some of the students are found to dropout because of lack of enthusiasm. By doing the research extensively some of the solutions proposed are: use of imperative programming language initially and gradually shifting to the object oriented approach. Use of analogies is suggested for teaching the fundamental concepts and student should see the relevance of what he is studying. Iterative approach to learning is suggested and laboratory-based model is better approach for teaching.

One-hour lectures are preferred over two-hour lectures and the laboratory sessions are merged with tutorial sessions. Mentoring sessions are arranged to provide one-to-one interaction. Apart from the regular tests, assignments are used for assessment. Game playing tasks are chosen as assignments to make them student friendly. Second year students are used for mentoring classes. An online submission system is used for assignments to provide instantaneous feedback.

The taxonomy of languages and environments designed to make programming more accessible to novice programmers of all ages is presented as in [11]. Learning programming has various social barriers. These barriers are to be addressed to make programming interesting to the students. Programmers are encouraged to work in a group to overcome social barriers.

III. Observations and Analysis

With the experience in teaching programming and research, the following are some of the suggestions for effectively teaching programming to first year under-graduate students:

The language chosen should be simple to learn but yet efficient. Procedure oriented programming languages appeal directly to the human thought process. Humans are accustomed to solving problems by following a procedure. Object oriented programming languages are intrinsically suitable when the problem to be solved is complex. Though object orientation is how we perceive the real world, this is not how we solve problems. Among the widely used languages C, C++, and Java, language which is more suitable for first year under-graduate students is C. Though language is essential for teaching programming, but more important is problem solving.

There is need for languages which are suitable for wide range of learners. The language chosen should be modular by design. It should have features which are independent subsets.

These different subsets can be learned depending on the requirement.

The language chosen should have demand in the market but it is not end in itself. Definitely it is difficult to get job based on the programming language studied at first year level. More advanced programming languages are studied in under-graduation which results in student having better job-prospects. Hence fundamental programming language which enables student to solve a variety of problems is appropriate.

Engineering is about problem solving. Programming is not to study the syntax and semantics of a programming language. Syntax and semantics are never the attractive features and doesn't result in enough enthusiasm for the student to study the subject. Student should understand the problem solving aspects first before attempting programming. In fact programming is just an application of problem solving. If student solves the problems on his own, even if the problems are simple, it can create enough enthusiasm and zeal for the student to concentrate on programming and feel at ease.

Thought should be given to the revision of syllabus of fundamental programming subjects also during syllabus revisions which should happen regularly. This often neglected aspect should consider the current trends which include advancement in teaching tools, student familiarity with programming before under-graduation. Framing the syllabus should be a continuous process and the feedback from all the stake holders should be taken whenever appropriate. Social and motivational aspects are to be considered while framing the syllabus. Even though the syllabus is revised regularly, implementation of it is not correct. Syllabus is revised and it is applied to the students who join the course in that year but syllabus is set for all four years. In professional engineering courses where changes are continuous, it is not appropriate. While applying the syllabus, student of any year should study the syllabus which is either latest or set one year back not more than that.

The teaching tools should be used effectively. Multiple tools are to be used depending on the requirement. Black board is effective for teaching different concepts. Wherever dynamism is required projectors have to be used. Different features provided by software tools are to be used effectively, particularly single step mode of tracing the program watching the change in values of variables during program execution.

Large size of class rooms prevent the teacher from concentrating on individual students. Class room's size has to be reduced. It should be made mandatory to conduct mentor classes or tutorial classes. They are to be conducted for the purpose for which they are designed not to conduct lecture classes because in majority of colleges they are converted to lecture classes.

Student has to prepare properly before he studies programming subject. Information regarding what the students study in the course they joined, what is the inter relationship between various subjects, what skills they learn, and what kind of job prospects they have in the market are to be conveyed clearly. In reality inter-relationship between subjects is never conveyed explicitly and information about job prospects is left to the final year of the course.

With the growth of Internet there is no meaning in depending on what is taught in the class room. Every student identifies his own time to study and when he is studying, if he gets doubts, those doubts should be clarified. A teacher can create a portal for the subject and can place all the material like lecture notes, audio and video presentations etc online. Student can effectively use them as per his convenience. Since some students are from rural background they don't have good communication skills and they don't ask questions (doubts) in the class room. These people can definitely ask questions online because of the privacy they get when using the Internet facility. Student should be made aware of other facilities like newsgroups etc.

Incremental programming often helps the students in learning programming in a step by step manner. If students are asked to write complete programs in the beginning of the course, they fail to do so and it de-motivates the student. Instead student should be allowed to debug small programs. Tools which enable the student to watch the execution of programs step by step can help him in understanding the programs. Tools which enable how the values of variables change during the step by step execution of the program will help a student a lot. Integrated development environments (IDEs) like Turbo C IDE provide such tools.

A human learns by using his previous knowledge. He always tries to relate new things to what he already knows. The interlinking between programming concepts and what the student already knows, results in effective learning.

Proper preparation for the class is essential for the teacher. This is often neglected particularly by senior teachers and this puts teachers in embarrassing situation. For young teachers preparing for the class includes learning new things, but for experienced teachers it is planning for the lecture psychologically. While preparing the plan we should put ourselves in student place and think from student perspective and not as experienced teacher. Also permit the student to come prepared for the class. Class room is not a place for surprises. Providing lecture notes to the student should not be considered as leaking the information, but it makes student study them, think about them and come to the class ready to learn new things from what he already knows.

The evaluation system should have two important goals. It should challenge the intelligent student but at the same time motivate the students who are relatively less intelligent due to social background, environment in which they are brought up etc. Continuous assessment particularly at initial stages puts the student on the right path. The evaluation should not be restricted to objective and essay type questions. Evaluation which permits the student to identify the errors in the program and correct them is a better choice for programming subject. Self evaluation is one of the better choices because the evaluation is done by him and it protects the privacy of the student. It enables the student to identify his own strengths and weaknesses. From the psychological perspective it is very effective because student agrees with his own assessment. Student often have many complaints about the assessment made by others, including teachers. But no-way he can disagree with his own assessment.

IV. CONCLUSION

Computer science has influenced all engineering fields and programming has become essential component of all engineering fields. Teaching computer programming has always remained a difficult task because of the dynamics of programming. Problem solving should be an integral part of computer programming and significant importance has to be given to it. Student should solve a variety of problems before writing even simple programs. Teacher and student should come prepared for the class, though the preparation required is different. Black board is convenient for explaining the concepts, the dynamics of a program can be presented easily with projectors, and tracing tools enable student to understand the working of a program. Continuous evaluation of student including self evaluation is essential to identify strengths and weaknesses of student and improve programming ability. All said well, learning and teaching programming always remains a challenging task.

REFERENCES

[1] http://www.ieagreements.org/washington-accord.

[2] Anthony Robins, Janet Rountree, and Nathan Rountree, "Learning and Teaching Programming: a review and discussion", Computer Science Education, Vol. 13, No. 2, pp. 137-172, 2003.

[3] OzgurAktunc, "A Teaching Methodology for Introductory Programming Courses using Alice", International Journal of Modern Engineering Research (IJMER), Vol.3, Issue.1, pp. 350-353, Jan-Feb. 2013.

[4] http://www.aicte-india.org/statistics.php.

[5] Peter J. Denning, "A Debate on Teaching Computing Science", Communications of the ACM, Vol. 32, No. 12, Dec. 1989.

[6] http://www.abet.org/accreditation-matters-students/

[7] http://nbaind.org/En/1066-washington-accord-signatory.aspx

[8] Jerome S. Bruner, "The Process of Education", Harvard University Press, 1999.

[9] Keiichitakaya, "Jerome Bruner's Theory of Education: From Early Bruner to Later Bruner", Interchange, Vol. 39, Issue. 1, pp: 1-19, DOI: 10.1007/s10780-008-9039-2, 2008.

[10] IwonaMiliszewska and Grace Tan, "Befriending Computer Programming: A Proposed Approach to Teaching Introductory Programming", Issues in Informing Science and Information Technology, Vol. 4, 2007.

[11] Caitlin Kelleher and Randy Pausch, "Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers", ACM Computing Surveys, Vol. 37, No. 2, pp. 83–137, June 2005.