Problem-based Learning using Simulation Tools for Automata Theory

Dadaso T. Mane [1], Sadanand S. Howal [2], Varsha T. Lokare[3]

[1, 2, 3] Information Technology, Rajarambapu Institute of Technology, Islampur, Sangli, Maharashtra

[1]dadaso.mane@ritindia.edu, [2]sadanand.howal@ritindia.edu, [3]varsha.lokare@ritindia.edu

Abstract: This paper highlights problem-based learning method implemented during the classes of automata theory for the second year through best practices activity. In this paper, we are providing conclusion based on the activities performed during academic years 2013-14 and 2014-15. These conclusions are made from the activities such as think-pair-share, group activity; problem based learning activity simulation tools and so on. We made comparison of the overall performance and satisfaction between the problem-based and traditional learners, and made analysis of the influence of students' mathematical knowledge in both methods. The results collected from these activities were tremendous; students went through the activities rigorously; they got clear understanding of the course conducted by their audit and grades during both years. This new experience was joyful for both the students and the teacher; and finally, the innovative teaching approach helped several types of learners very well.

Keywords: Innovative teaching, Problem-based learning, simulation tools, best practices activity

Dadaso T. Mane
Information Technology, Rajarambapu Institute of Technology, Islampur, Sangli, Maharashtra
dadaso.mane@ritindia.edu

1. Introduction

The theory courses of information technology stream are an important but challenging one. The course automata theory has been found very difficult to the students. According to student feedback, the major reason for the dislike is the mathematical and theoretical nature of the topics as well as complex notations used in the course.

Even with more difficult topics to learn deal with problems which are computationally unsolvable, i.e. there is no algorithm to describe the method to be learnt. Some topics such as the Pumping Lemma for regular languages (how to prove a formal language non-regular) or Turing Machines are classical examples. In addition, the traditional material for the topic does not introduce any practical applications or focus the meaning of the issues in real-life problems of computer science. The main concept behind *problem-based learning* (*PBL*) is to use problems, questions or puzzles as a starting point for learning. PBL is thus a student-centred method which matches well with the *constructivist* view of learning [1]: it requires active processing of information, activates learner's prior knowledge, stimulates elaboration and organization of knowledge, and offers a meaningful context for learning.

While going with *problem-based learning* (*PBL*) our aim was to test the following hypotheses:

1. Mathematical background has a strong influence on student's success in automata theory.
2. Problem-based learning attracts more female students. [2]
3. Average students manage better in problem-based learning.
4. Problem-based learning prevents dropping out.

When we started to design a new approach for automata theory course, our first goal was to change the role of students to be more active. With difficult courses it is especially crucial that the students will not remain just as passive receivers of new information, but they should

become active constructors of the new knowledge, as the constructivist hypothesis argues [3].

2. Course Description

The traditional method of teaching for the course automata theory has average impact upon students. In this method, teacher first introduces concept and solves problems on blackboard. Here, students are asked to solve examples by following method.

1. Draw the state transition diagram of computational model on blackboard.
2. Explain working principle of this model work as per formal definition.
3. Explaining how this computational model processes the string by writing the sequence of transition (while processing string) on the board.
4. At the end concluding of recognition of the language.

This teaching method has several drawbacks such as:

1. It is very time consuming process to draw design models of examples as well as to write sequence of transitions on the board.
2. It is not possible for the teacher to cover several examples in the class at a time.
3. Students lack in proper understanding of the concepts and the working principles of models and hence, not able to draw the computational models for any given problem statement.
4. Slow learners have to face lot of problems with this method.
5. Students are not able to recall definitions of these models and not able to differentiate among working principles of different computational models

To overcome all of the above mentioned issues, we decided to go with new teaching method for the betterment of students learning. During the last year, we have implemented the problem-based learning method along with open source simulation tools such as JFLAP [4] and JFAST for teaching automata theory course [5].

3. Course Implementation

The course covers the theory of computability from finite automata and regular expressions to context-free grammars, pushdown automata, Turing machines and solvability issues. The course was having 3 lectures per week and a tutorial. The course teacher would start off with introduction of the concept and explanation by solving example on the black board as well as its demonstration using advanced IT tools such as JFLAP [6] and JFAST in the first half of the lecture time. In the remaining lecture time, students were asked to solve given examples on the black board in the class and to demonstrate the same using JFLAP and JFAST software's .The simulation tools were used for demonstration of problems are explained with example as below:

A. JFAST Tool

This is very simple and user friendly GUI tool to design DFA, NFA and NFA-^. This tool has been used to design as well as simulation of given example. It provides complete working of DFA, NFA and NFA-^ over a given example. The following example shows how we can use JFAST to design and simulate DFA [7].

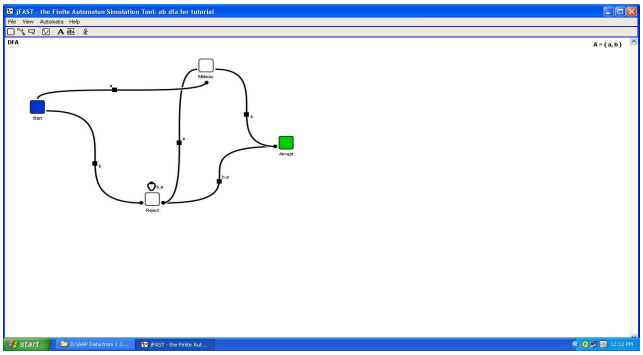


Fig. 1 JFAST GUI

B. JFAP Tool

JFAST don't have functionality to draw PDA and TM. So, we have used JFALP to design PDA. We can see simulation and working of PDA designed using JFLAP.
Following example shows how we can use JFLAP to design and simulation of PDA.

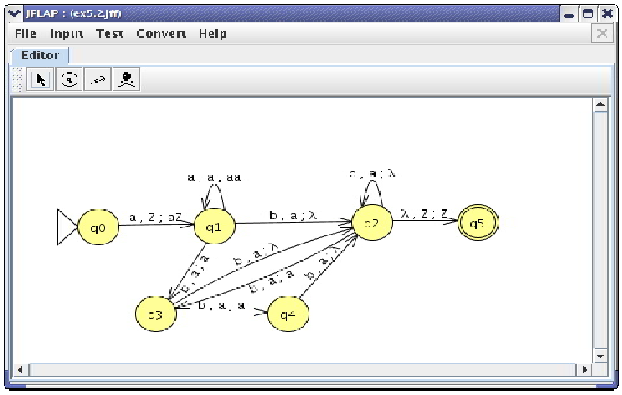


Fig. 2 JFLAP GUI

The following table shows a plan of course implementation for automata theory.

Table 1. Plan of course implementation

Lecture No.	Tool to be used	Particulars
1- 5	Think pair share activity	Prediction of Regular Expression(s)
6-10	JFAST tool	Designing and simulation of DFA.NFA and NFA-^
11-18	Problem solving activity in group with presentation.	Conversion of NFA-^ to NFA to DFA
19-23	Think pair share activity	Writing CFG.
	Problem solving activity in group	For conversion of CFG into CNF format.
24-31	JFLAP tool	Designing and simulation of PDA
	Problem solving activity in group with presentation.	Parsing.
32-34	Video lecture(s)	Basic working of TM.
	JFLAP tool	Designing and simulation of TM

We planned activities taking problem based learning broadly along with use of design and simulation tools like

JFAST and JFLAP. This process helped to students and teacher to clear understanding of concepts. This made class room teaching and learning very effective and joyful.

4. Course Assessment

Automata theory course has been offered in autonomous curriculum of second year class where we emphasizes on analytical and problem solving skills of students. The course has defined certain course outcomes for students upon completion. The course outcomes are aligned with the program outcomes of Information Technology.

Automata Theory course has following course outcomes where it is expected that after completing this course successfully, students should be able to:

1. Predict the Regular Expression for given language.
2. Design DFA, NFA for given language
3. Construct the context free grammar (CFG) for given language.
4. Parse the given input using top down & bottom up parsing.
5. Build the Turing machine for various types of languages and use the Turing machine for computation of function.

We have evaluated Course Outcomes (COs) attainment based on marks obtained and feedback by students during academic years 2013-14 and 2014-15.These methods are described as follows:

A.CO-attainment by marks obtained:

We have applied formal assessment method which generally refers to the regular assessment that is used in evaluating students for their grades obtained in the course. In this method, we have used existing data from student's marks, for example from the test results, final examination, quizzes and problem solving contests. In autonomous curriculum, we have three examination components: In Semester Examination (ISE), Mid Semester Examination (MSE) and End Semester Examination (ESE). The weight ages for these components are 20%, 30% and 50% respectively.

ISE is conducted using Quiz (5%) and problem solving contests (15%) during tutorials.

In general, the following formula is used for CO attainment [8]:

$$\text{AttCO}_i = \sum_{E=1}^{N} \frac{\sum_{Q=1}^{K}(\text{AvgM})i}{\sum_{Q=k}^{K}(\text{TotalM})i} \times \text{Weight}_E$$

……formula (1)

Where i= Total number of Course Outcomes

N= Total number of exams conducted

E= Type of Exam (ESE, MSE, Quizzes, Problem solving contest)

Avg. M (Q1 to k) i = Average Marks obtained by students in question 1 to k (where questions 1 to K associated with COi)

Total M (Q1 to k) i = Total Marks assigned to question 1 to k (where questions 1 to K associated with COi)

We have evaluated CO attainment by the formula (1) for automata theory as follows:

AttCO = AvgMarksESE×0.5 + AvgMarksMSE ×0.3+ AvgMarksQuizzes×0.05+ AvgMarksPSC×0.15

Here we have considered 160 students marks obtained in academic years 2013-14 and 2014-15 to measure CO attainment of Automata Theory course. At RIT, we have feedback system where we conduct course exit survey for CO attainment. We conduct course exit survey based on certain questionnaire which is mapped with COs of respective course. Table 1 and Table 2 show the results obtained by applying formula (1).

Table 2: CO Attainment in academic year 2013-14

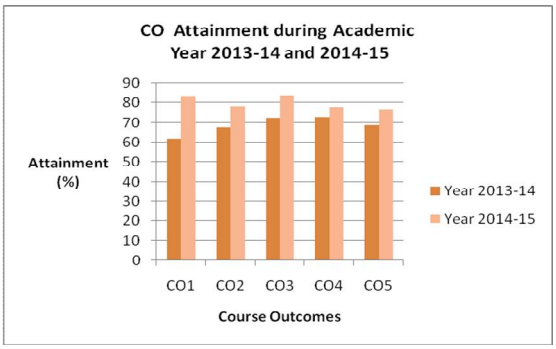
Assessment		Marks	Overall % age	Course Outcome Attainment (in % age)				
				CO1	CO2	CO3	CO4	CO5
ESE		100	50	75.5	72.5	82.5	72.5	71.9
MSE		50	30	72.4	62.2	67.5	77.5	62.5
ISE	Quizzes	20	05	05	69.8	65.3	66.8	65.2
	Problem Solving Contest	20	15	15	75.3	70.6	72.9	77.1
Average				61.97	67.6	72.4	73.0	68.7

Table 3: CO Attainment in academic year 2014-15

Assessment		Marks	Overall % age	Course Outcome Attainment (in % age)				
				CO1	CO2	CO3	CO4	CO5
ESE		100	50	82.5	80.5	77.2	88.1	74.5
MSE		50	30	77.3	67.3	88	72.4	82.4
ISE	Quizzes	20	05	86.2	82.2	83.5	77.8	79.8
	Problem Solving Contest	20	15	88.1	83.4	87.2	72.5	70.3
Average				83.5	78.4	83.9	77.7	76.7

We have plotted graph based on above statistics to show the comparative analysis of CO attainment.

Graph 1: CO attainment by marks obtained



B.CO Attainment by Feedback:

Here, we have made a questionnaire for conducting course exit survey to map to corresponding COs and responses generated by students have been considered to measure the attainment by feedback method. We have applied following formula:

$$AttCoi = \sum_{Que=1}^{N} \left(\frac{\sum_{i=1}^{3} WOpti \times NumOfstudi}{TotalNoofStud} \right) \ldots \text{formula (2)}$$

Where, i = No of options (Excellent, Very good, Good, Average, Poor etc.)

$WOpt_i$ = Weight assign to each option i (e.g. Excellent (10), Very good (7), Good (5), Average (2) and Poor (0))

$NumOfstud_i$ = No. of students responding to respective option i

Que=1 to N: Questions associated with each COi

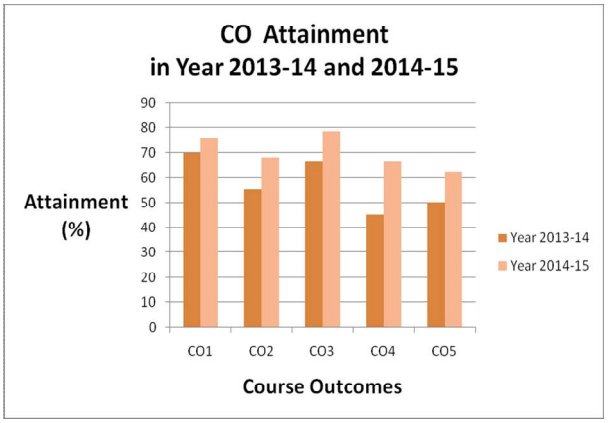
By applying formula (2) we got the following result:

Table 4: CO Attainment in academic years 2013-14 and 2014-15

Course Outcome					
	CO1	**CO2**	**CO3**	**CO4**	**CO5**
2013-14	69.71	55.0	66.2	45.0	50
2014-15	75.2	67.8	78.3	66.3	62

We have plotted graph based on above statistics to show the comparative analysis of CO attainment.

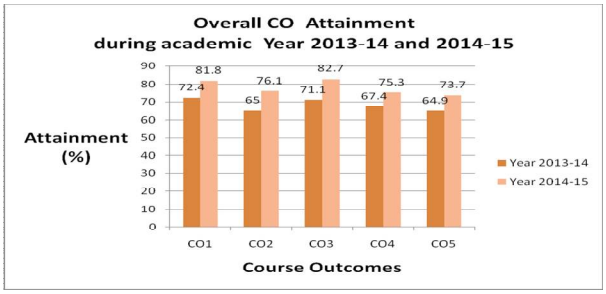
Graph 2: Co attainment by feedback Method



C. Overall CO Attainment:

Finally, we have evaluated the overall percentage of CO attainment, where we have given 80% weightage to first method which is based on marks obtained and 20% weightage to second method which is based on feedback. Hence, the overall CO attainment is shown as below.

Graph3: Overall CO attainment for academic years 2013-14 and 2014-15



Graph 3 shows the CO attainment in percentage for academic years 2013-14 and 2014-15. This course has total five course learning outcomes. From the graph, the observation is that CO attainment for an academic year 2014-15 has been significantly increased as compared to an academic year 2013-14. As we have implemented problem based learning using simulation tools JFLAP and JFAST during an academic year 2014-15, it shows that PBL method helps students to better learning of Automata Theory course.

5. Conclusion

Problem based learning along with open source simulation tools has great impact upon teaching learning process in classroom delivery. Our aim of undertaking this method was students should be able to grasp and understand the formal definitions, notations used in Automata Theory. Students should be able to design and differentiate various computational machines such as PDA, DFA, and Turing Machine. This may help students to succeed in competitive examinations like GATE and various professional certifications in Information Technology. The results we have obtained from the implementation of this teaching learning method are quiet encouraging and successful. From overall Course Outcome (CO) attainment comparison for this course statistically makes clear that, new approach has provided better opportunity to learn and make classes interactive both for teacher as well as students.

References

1. Ben-Ari, M., 2001. Constructivism in computer science education. Journal of Computers in Mathematics and Science Teaching 20 (1), 45–73.
2. Lambrix, P., Kamkar, M., 1998. Computer science as an integrated part of engineering education. In: Proceedings of the 6th annual conference on the teaching of computing and the 3rd annual conference on integrating technology into computer science education. ACM Press, pp. 153–156.
3. M. Ben-Ari, “Constructivism in computer science education,” Journal of Computers in Mathematics and Science Teaching, vol. 20, no. 1, pp. 45–73, 2001.
4. http://www.jflap.org/tutorial/
5. Rodger, Susan H. Visual and Interactive Tools. Web site of automata theory tools at Duke University, Feb. 2005. Available online at: http://www.cs.duke.edu/~rodger/tools/.

6. R. Cavalcante, T. Finley and S. Rodger. A visual and interactive automata theory course with JFLAP 4.0. In Thirty-fifth SIGCSE Technical Symposium on Computer Science Education, pages 99-99, ACM Press, 2004.
7. Timothy M. White and Thomas P. Way "jFAST: A Java Finite Automata Simulator" SIGCSE'06, March 1-5, 2006, Houston, Texas, USA. Copyright 2006 ACM 1-58113-997-7/05/0002...$5.00.
8. Izham Zainal Abidin, Adzly Anuar and Norshah Hafeez Shuaib, "Assessing the attainment of course outcomes (CO) for an engineering course", in International Conference of Teaching and Learning (ICTL 2009) INTI University College, Malaysia.