CURRICULUM DESIGN: AN EXPERIENCE IN PRINCIPLES OF COMPLIER DESIGN COURSE

Sujatha C[1] , Karibasappa K. G[2]
Dept of Computer Science and Engineering
B.V.B.College of Engineering and Technology
Hubli, Karnataka, India
[1]sujata_c@bvb.edu
[2]karibasappakg@bvb.edu

***Abstract*— Curriculum design is a challenging process which includes content design, delivery of course, learning and assessment methods based on the need of different stakeholders. The content design is a process of collecting, organizing and sequencing the lessons or content to study the course in orderly manner. It should recognize students as an active partner in their learning process. Generally it is concerned with reviewing, planning, developing, implementing and maintaining of course content design, delivery of course and assessment with high level commitment towards the curriculum by stakeholders. In this paper we present our experience in the content development, delivery and assesment of the Principles of Complier Design (PCD) course.**

Keywords— Compiler, Curriculum, Content Design, Analysis phase, Moodle assessment tool.

I. INTRODUCTION

Our previous curriculum of Computer Science and Engineering had three Language processing related courses namely System Software, Compiler Design and a prerequisite course Finite Automata and Formal Languages (FAFL) for Compilers in different semesters. The language processing system is divided into two parts: one is front end(machine independent) and back end(machine dependent) as shown in Fig 1.1. The figure depicts that compiler need to be taught before teaching the assemblers, loaders and linkers. But in the earlier curriculum the courses were taught in the reverse order. Looking at these issues we propose to combine the three courses into two courses, one as PCD and another as system software. The PCD course deals with the prerequisites needed for compilers i,e FAFL concepts & its applications i,e the front end of compiler (lexical, syntax & semantic analyzer, intermediate code generator called as analysis phase) and the system software deals with backend of compiler(synthesis phase), assembler, linker & loader. Accordingly we proposed to change the content and delivery approach. In this approach, we teach the prerequisite and then followed by its application i,e the phases of compiler that belong to front end of compiler. Here our main focus is to teach compilers as an application of FAFL concepts. For eg we teach regular expressions followed by its application i,e recognition of tokens (lexical analyzer).

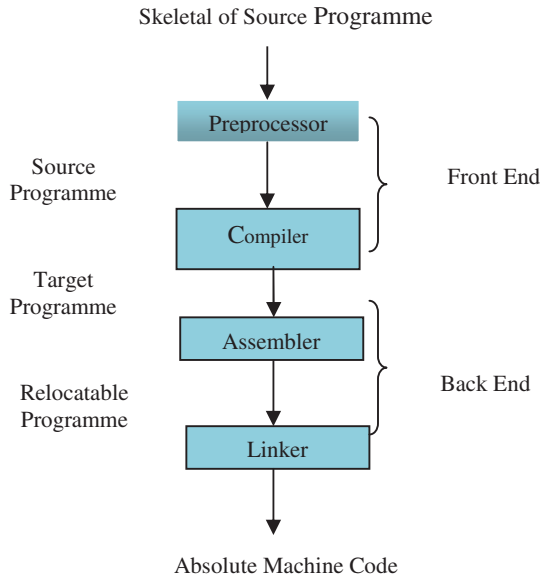

Fig.1.1: Language Processing System

II. LITERATURE SURVEY

ZhaoHui[1] have discussed the problems in teaching principles of compiler and proposed solutions based on student centric teaching approaches such as concept mapping, problem based learning (PBL)[2], case study and e-learning and their effectiveness in learning the course.

LISA[3] is an integrated development environment in which students can specify, generate, compile-on-the-fly, and execute programs in a newly specified language. Students can experiment, test and analyze front end compiler phases such as lexical and syntax analyzer along with attribute evaluation strategy. LISA help the students to understand the basic concepts of compilers in more efficient and direct way.

Marjan Mernik, Viljem Žumer [4] discussed the utilization of software tool called LISA (Language Implementation System Based on Attribute Grammars) for project based learning of compiler construction. Authors expressed the benefits such as constructive learning, active learning, learning speed and better understanding of concept in using LISA tool.

Wang na and Zhang Shi Ming [5] have integrated compiler technology and compiler theory and experimented the combination of manual and automatic methods for

constructing lexical and syntax analyzer using Lex and Yacc tool.

This paper presents experience of integrating finite automata, front end of compiler and the implementation of lexer and parser. We have used some pedagogical activities to enhance the teaching learning process.

III. COURSE DESIGN

System software related courses are the important components of Computer Science and engineering stream of U.G. Programme. Courses typically deal with concepts such as, lexer, parser, code generator, optimizer, assembler, loader and linker. These concepts are organized in the language processing system[6] as shown in the Fig. 1.1. These concepts require the knowledge of automata theory which is used in the different phases of compiler. In the previous curriculum these concepts were taught in different courses at various levels (semester) in the isolated way as shown in Fig 1.2..

Fig.1.2: Previous curriculum.

The isolated way of teaching lacks in conceptualizing and better understanding the whole language processing system. This drawback and swebok [7] survey motivated us to revisit the content design of system software courses.

The content design of PCD Course in Computer science Curriculum is based on the concept mapping with the other courses as shown in the Fig.1.3 Preconceptions required for PCD are taught at the lower semesters and few courses are taught in the same semester.

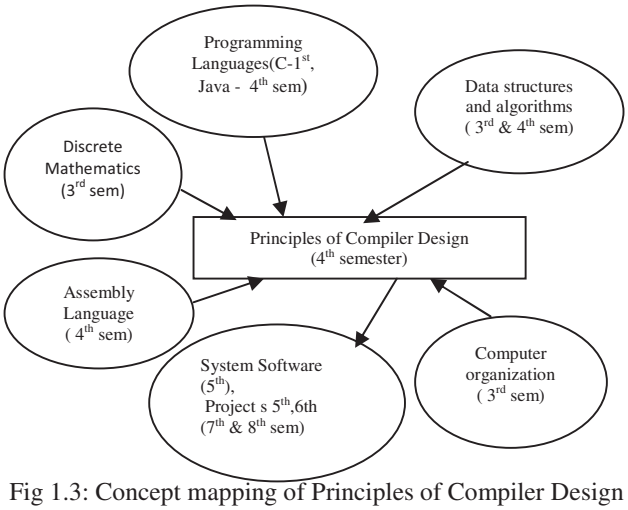

Fig 1.3: Concept mapping of Principles of Compiler Design with other courses in the curriculum

The new curriculum combines the Automata theory and compiler design concepts into one course named as principles of compiler design(PCD) as shown in Fig. 1.4.

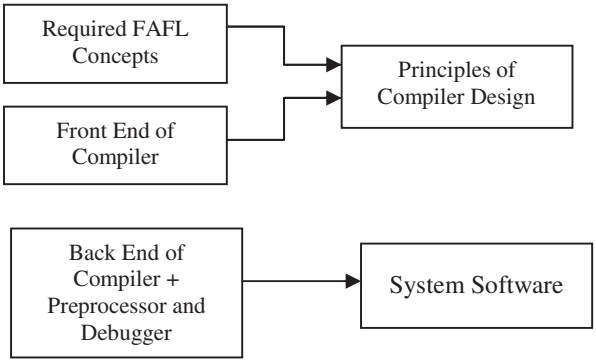

Fig.1.4: Revised and Existing Curriculum.

The PCD course consists of prerequisites for compilers i,e FAFL concepts & front end of compiler (lexical, syntax & semantic analyzer, intermediate code generator called as analysis phase) which is five credit course. Out of which one credit course is fixed for self study(SS) component. The flow of course content is organized as, the required pre requisites from automata theory are followed by the respective phases of the compiler. For example regular expression and deterministic finite automata are discussed followed by the compiler phase called lexical analyzer as an application of regular expression. This flow helps the students to easily understand and better learning of concepts. Course content design & delivery is discussed in the following section.

IV. CONTENT DESIGN AND DELIVERY

The approach followed in current content design is that, the prerequisites of compiler phases are followed by their applications i,e front end of compiler phases. For eg the regular expressions are followed by its application i,e recognition of tokens (lexical analyzer).

The PCD course is a five credit theory course, where four credits are for the course content and one for the SS component. As there is no associated lab, we also give the practical exposure through the self study comopnent. The flow of course content design and delivery is shown in Fig. 1.4.

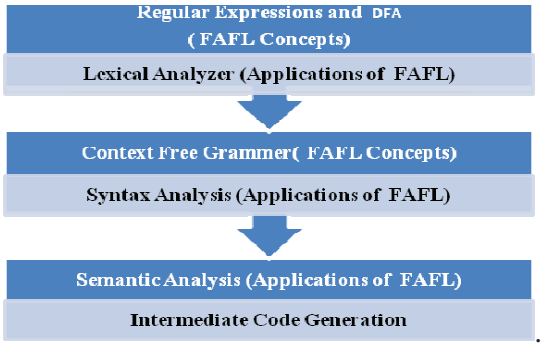

Fig. 1.4:Concept map based Content organisation and Flow of delivery

As the course is one of the challenging subject of CSE, we want the students to have a regular and continous study of

course concepts. Hence we used few pedagogical activities such as framing of Objective type questions of various patterns chapter wise, quizzes, survey of compilers & tools, implementation of lexer & parser using tools and C implementation of parsers. Framing of questions, survey and implementaions were the part of SS component. The activities were in group(4-5 students in a each group), but the quiz was an individual activity conducted using Moodle assesment tool. Table 1 shows some samples of activities carried out on each topic and the program outcome (PO) mapped[8].

These activities ensured the students learning and better understanding of the concepts. It also gave an opportunity to appreciate the automata concepts by taking the examples of existing languages such as C rather than hypothetical languages in the prerequisite course (FAFL). We address the PO c for implementing the top down and bottom up parsers ,PO i for self learning to know about the various compilers and tools used to build the compilers, PO k for knowing and using the tools which was the extra knowledge gained. All these activities were the part of SS component.

V. ASSESSMENT

A. Continuous Internal Evaluation(CIE)

Apart from the regular assessment in minors & semester end exam, the students were assessed through different pedagogical activities. All the activities except quiz were assessed based on the evaluation parameters and rubrics. As the quiz was conducted using the Moodle assessment tool the analysis become easier. To conduct such online quizzes we need a huge set of questions. This motivated us to prepare question bank chapterwise. An attempt was made to give this task as one of the group activity in SS component. Chapterwise question bank of different types as mentioned in Table 1 along with solution was created and submitted by the team.

Topic	Activity	Sample	PO addressed
Regula Expression and Finite Automata	• Concept Mapping • Objective type Question framing. • Quiz	Different Question types: Multiple choice,fill in the blanks, match the nearest, True/false, complete the missing part, reasoning	i, k, c
Lexical Analyzer	• Concept Mapping • Survey on compilers & tools • Implementation lexer & parser using tool • C implementaion of parsers • Quiz	Compilers for different programming paradigms: functional, logic,object oriented, procedural, scripting, fourth generation etc. Recognize C and C++ tokens	
CFG	• Concept Mapping • Objective type Question framing. • Quiz	Different Question types: Multiple choice,fill in the blanks, match the nearest, True/false, complete the missing part, reasoning	
Syntax Analyzer	• Concept Mapping • Survey on compilers & tools • Implementation parser using tool • C implementaion of parsers • Quiz	Implement to validate arithmetic expression using LL(1) parser.	
Symantic Analysis	• Concept Mapping • Objective type Question framing. • Quiz	Different Question types: Multiple choice,fill in the blanks, match the nearest, True/false, complete the missing part, reasoning	
Intermediate Code generation	• Concept Mapping • Objective type Question framing. • Quiz	Different Question types: Multiple choice,fill in the blanks, match the nearest, True/false, complete the missing part, reasoning	

Table 1: Chapter wise pedagogy Activities and POs addressed

The questions & solutions were reviewed and modified if required by the factuly. This activity not only made the students to study regularly, clarify the concepts but also hepled to build the question bank which could also be used in preparation for competative exams. The moodle tool helped us to analyze the scores of students and also analyze the quality of questions[9].

The Fig 1.4 (a) & (b) shows the number of students acheiving grade ranges and statistics for each question. The graph (a) shows the sample of grade ranges scored by set of students. In the graph (b) the two parameters facility index and discriminate efficiency help to anayzle the questions. *Facility index is* the mean score of students on the item (question).

As per the [9] the percent range 11-20 indicates difficult question, 35-64 indicates a right question for the average student,66-80 is fairly easy question. From the graph b) we observe that 9(36%) questions are average, 12(48%) are fairly easy and 4(16%) are difficult questions. The discriminate efficiency statistic attempts to estimate how good the discrimination index is relative to the difficulty of the question.

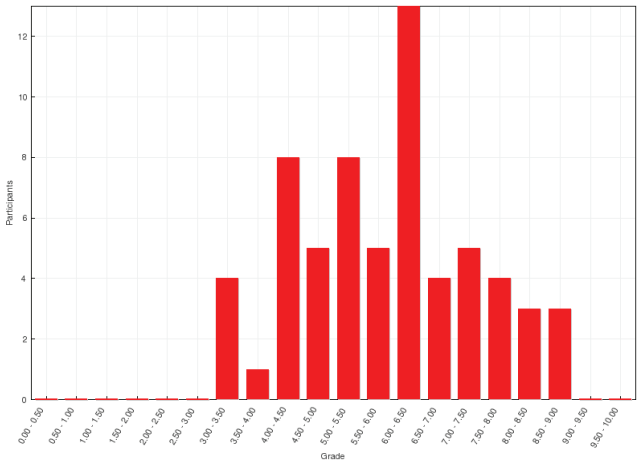

Fig. 1.5.(a) # students achieving grade ranges

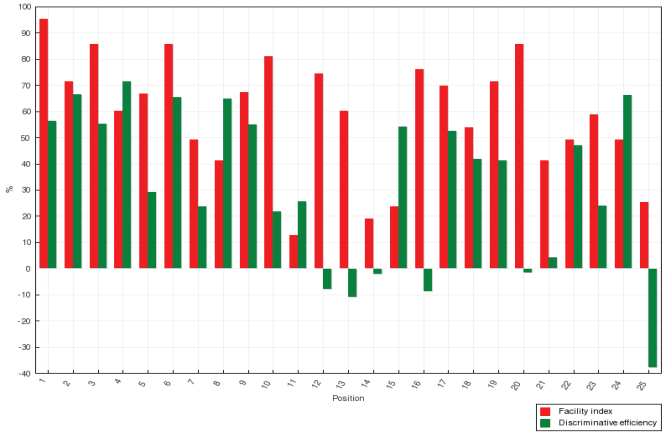

Fig 1.5.(b) Statistics for question positions

Discrimination index is the correlation between the weighted scores on the question and those on the remaining. It indicates how effective the question is at sorting out able students from those who are less able. A question which is very easy or very difficult cannot discriminate between students of different ability, because most of them get the same score on that question. Maximum discrimination requires a facility index in the range 30% - 70%. From the graph b) we observe that 14(56%) is very good discrimination, 5(20%) is adequate. Other 6(24%) is weak which indicates difficulty level of question is more. These statistics can help the faculty to grade the quality of question paper.

B. Semester End Exam. (SEE)

We have analyzed the results of compiler course and automata theory taught in isolated for 2011-12 batch, whereas the integrated syllabus of automata theory and front end of compiler taught in 2012-13 & 2013-14. In the current batch the change in content flow and pedagogical activities were adopted.

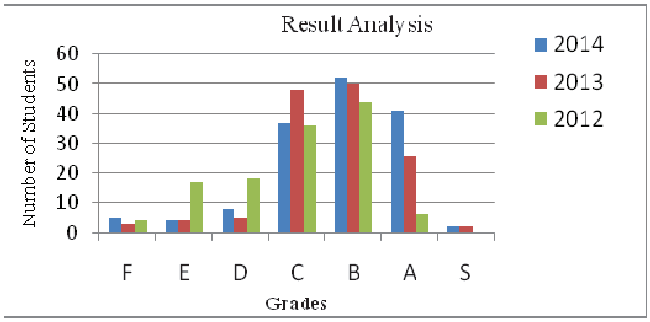

Fig. 1.6: Comparison of Result Analysis of previous three years

From Fig.1.6 we observe that results in 2013-14 have more number of A grades and B grades as compared to 2011-12 and 2012-13. The integrated approach and practice of pedagogical activities helped in achieving the good results.

VI. CONCLUSION

Authors combine the automata theory concepts and analysis phase of compiler as a PCD course. Here our focus is on the application of automata theory. We have practiced few pedagogical activities that ensured students learning and better understanding of the course, the practical exposure also added to the knowledge gained by the students. The Moodle tool made us easy to conduct, evaluate and analyze the quizzes and questions which help to improve the teaching & learning process. The quiz statistics helped us to analyze the students scores and the quality of question paper.

References

[1] Zhaohui Li, " Exploring Effective Approaches in Teaching Principles of Compilers," The China Papers Nov 2006.

[2] Khoumsi, A. and Gonzalez-Rubio, R. "Applying a Competency and Problem-Based Approach for Learning Compiler Design," Journal of STEM Education, Vol **7** , issues 1,2, 2006.

[3] Mernik Marjan, Lenič Mitja, Avdičausević Enis, Žumer Viljem, "LISA: an interactive environment for programming language development," Horspool N. (ed.). 11th International Conference Compiler Construction, LNCS No. 2304, pp. 1-4, 2002.

[4] Marjan Mernik, Viljem Žumer, "An Educational Tool For Teaching Compiler Construction," IEEE Transactions On Education, Vol. 46, No. 1, February 2003.

[5] Wang na and Zhang Shi Ming, "The Progress of Construction of Compiling technology course based on different methods", Third International conference on Science and Social Research ICSSR-2014

[6] Alfred V Aho, Monica S. Lam, Ravi Sethi, Jeffrey D Ullman: Compilers - Principles, Techniques and Tools, 2nd Edition, Pearson, 2007.

[7] ISO/IEC TR 19759:2005, Software Engineering - Guide to the Software Engineering Body of Knowledge. International Organization for Standardization, Geneva, 2005.

[8] National board of Accreditation New Delhi-110003: Manual for Accreditation of Undergraduate Engineering Programs. I edn .

[9] http://docs.moodle.org/dev/Quiz_report_statistics

APPENDIX

Sample Questions framed by the students

- Match with appropriate choices

Memory	PushDown Automata
Intermediate Node	Nonterminals (Variables)
S->(V U T)*	Production
S->B	Unit Production

- Which of the following statement is true?

A. SLR parser is more powerful than LALR parser.

B. LALR parser is more powerful than Canonical LR parser.

C. Canonical parser is more powerful than LALR parser.

D. The parsers SLR, Canonical LR & LALR have the same power.

- Let G be the grammar of the language L, if G does not derive any terminal the language is called

A Ambiguous language

B Null language

C Unambiguous language

D Formal Language

- Set of prefixes of right sentential forms that can appear on the stack of a shift reduce parser are called

A. Viable prefixes

B. sentential forms

C. handles

D. none of the given options

- Regular expression 'r' may contain characters that have special meanings. Such characters are called_____.

- An_____ is a transition that may occur without consuming any input characters.

- Regular expression for all strings of lower case letters that either begin or end in 'a' is

a) a[a-z]*a b) a?[a-z]*a
c) a[a-z]*a? d) a?[a-z]*a/a[a-z]*a?

- Consider a grammar S→(S) | a. Let the number of states in SLR(1), LR(1) and LALR(1) parser for the grammar be n1,n2 and n3 respectively. The following relationship holds good.

A. n1<n2<n3

A. n1≤n3<n2

B. n1=n2=n3

C. n1≥n3≥n2

- For the assertion and a reason given below. Indicate your answer by choosing one of the options below.

Assertion: Finite automaton can't check whether the string is a palindrome.
Reason: Finite automaton has finite number of states and the capacity to remember arbitrarily long amount of information.

#	Assertion	Reason
A.	True	True
B.	True	False
C.	False	True
D.	False	False