An Experience Report on Teaching Compiler Design Concepts using Case-Based and Project-Based Learning Approaches

Divya Kundra  
Deen Dayal Upadhyaya, India  
divy1395@iiti.ac.in

Ashish Sureka  
ABB Corporate Research, India  
asish.sureka@in.abb.com

Abstract—Compiler design is a course that discusses ideas used in construction of programming language compilers. Students learn how a program written in high level programming language and designed for humans understanding is systematically converted into low level assembly language understood by machines. We propose and implement a Case-based and Project-based Learning environment for teaching important Compiler design concepts (CPLC) to B.Tech third year students of a Delhi University (India) college. A case is a text that describes a real-life situation providing information but not solution. Previous research shows that case-based teaching helps students to apply the principles discussed in the class for solving complex practical problems. We divide one main project into sub-projects to give to students in order to enhance their practical experience of designing a compiler. To measure the effectiveness of case-based discussions, students complete a survey on their perceptions of benefits of case-based learning. The survey is analyzed using frequency distribution and chi square test of association. The results of the survey show that case-based teaching of compiler concepts do enhance students skills of learning, critical thinking, engagement, communication skills and team work.

Index Terms—Case-Based Learning, Cognitive Apprenticeship Model, Constructivism, Didactic Teaching, Problem-Based Learning, Project-Based Learning, Teaching Compiler Design.

I. RESEARCH MOTIVATION AND AIM

A majority of engineering classes involve traditional lecture-based approach in which learning is considered as oriented from teachers to students. The traditional teaching is concerned with teacher being the active controller, having the entire power and responsibility of the environment [1]. Case-based learning is different from traditional learning in the manner that it places students as the center of education process. Students are given importance in what and how they are learning. Students apply the theoretical knowledge in solving practical world problems in a supportive environment [2]. Application of case-based learning is useful in learning about compiler design concepts as it makes the learning easier and interesting. Through the cases developed from real world software, students understand and practice how things work in real world. Cases help to build skills like confidence and critical thinking. Also to our best of the knowledge, not much work has been done in teaching compiler design using case-based teaching methodology. Thus cases developed can be shared and used by other faculty while teaching compiler design course. Since compiler design course involves element of programming, writing a compiler by self can give students experience of large scale application development. Thus programming projects needs to be included in the course contents.

II. RELATED WORK AND RESEARCH CONTRIBUTIONS

Some tools have also been built to facilitate learning of compilers. In [3] authors have build a tool LISA that supports learning and conceptual understanding of compiler design in an efficient, direct and long lasting manner. [4] introduces a set of tools designed and improved for compiler design educative projects in C++. In [5] Chirp- a language specification and compiler implementation is proposed. [6] suggests that building a compiler for domain specific language (language specially designed for some specific problem) can engage students more than traditional compiler projects. In [7] authors argue that compiler teaching through an unusual programming language textttklx with target processor as the postscript interpreter is a good choice for teaching compilers. [8] presents an idea of integrating real compiler code into teaching theory of compilers.

In context to previous existing work, our paper makes following novel research contributions:

1) We developed cases for teaching core compiler concepts.
2) We propose and implement a complete teaching framework CPLC which contains learning objectives, case-based and project-based pedagogy and measures students understanding.
3) We evaluate the performance and impact of the proposed case-based pedagogy and demonstrate its effectiveness.

III. LEARNING FRAMEWORK: CPLC

We propose a learning framework CPLC- Case-based and Project-based Learning environment for teaching Compiler design concepts. We apply CPLC while teaching the analysis phase of compiler i.e. lexical and syntax analysis. The content in these phases is very important and is sufficient for learning many of the basic principles of a compiler. The model uses cases that are based on practical problems giving students hands on experience in solving complex real world problems. The model also involves giving a mini project at the conclusion of a case discussion. Projects for designing
Fig. 1: Learning Cycle showing Learning Activity, Learning Actions and Learning Strategies.

partial compilers using LEX (Lexical Analyser) and YACC (Yet Another Compiler Compiler) \(^1\) for different languages like C, Java, HTML, SQL, MATLAB and Python are given to students. Our proposed model CPLC is complete in itself providing learning objectives, a teaching methodology and evaluation of students understanding. CPLC learning model is combination of the 4 pedagogical models- Didactic Teaching [1], Problem-Based Learning [9], Cognitive Apprenticeship Model [10] and Project-Based Learning [11].

We propose and apply the framework shown in Figure 1 for teaching the lexical and syntax analysis phase of compiler design. The foremost learning activity consists of didactic teaching which introduces and discusses concepts of compiler design. For the concepts discussed in class, instructor explores new ideas and viewpoints and finds analogies in real life to develop interesting cases. Once a concept is covered in the class, we allocate the case to a team of 3-4 students. Students retrieve similar problems and concepts taught in class to do problem solving. Articulation [10] through discussions within a team help students to consider different point of views, understand problem better and come up with the best solution. Instructor coaches students by monitoring their activities, assisting and supporting them whenever necessary. Student reflects over his performance by self analysis and self assessment [10]. The results of reflection [10] are put into verbal form in form of presentations. Students are encouraged to create and ask inquiries resulting in inquiry-based learning. Team submits solutions in form of a written report to the instructor. Instructor provides in depth analysis of the team's performance. After completion of the case, a project is assigned to the team. Scaffolding [10] happens where the instructor guides and advises students about the projects so that they can cope up with the task situation. At the end project presentation takes place where students demonstrate their results.

**Sample Case Studies**

We developed multiple cases for lexical and syntax analysis phase. The cases have been publicly shared\(^2\) so that they can be used by other instructors of compiler course. We present summary of one of the case of lexical and syntax analysis.

**A. Case of Spam Detection (Lexical Analysis)**

Developers of an upcoming email service - mails.com want to make a spam filter that automatically detects and removes spam. The filter would consists of thousands of pre-defined spam-rules against which the email content will be compared. Anything matching to the spam-rules would categorize to be a spam component. The developers know that as spam filters evolves to better classify spam, the spammers will adapt their writing methods to avoid detection. Thus to build effective rules, the developers of mails.com begin to observe what kind of spam attacks can occur on filters. Example as statistical spam filters begins to learn that word like “offer” mostly occur in spam and starts to think “offer” as spam-rule, spammers began to obfuscate them with punctuation, such as “o.f.f.e.r”. Some of the other attacks are also explained in the case. Observing the attacks discussed in the case and reasoning what other attacks can occur, appropriate tokenization mechanism is to be decided to achieve maximum accuracy of the filter.

The challenges for the students in this case are:

1) Identify various tokenization attacks that can occur on spam filter.

\(^1\)http://dinosaur.compilertools.net/

\(^2\)https://github.com/Divya-Kundra/Case-Based-Teaching.git
2) Decide the most promising tokenization techniques that can be proposed for the system.
3) Evaluate the reliability of the proposed tokenization scheme by proving how it will be resilient to the attacks.

B. Proposed solutions to the case

Each team studied the case in the allocated time, discussed within itself the possible solutions and presented them. Some teams argued that tokenization attacks which include splitting or modifying key word features (using more of capitalisation or punctuations within the word) are most common and thus proposed solutions for them. Some presented obfuscation attacks (changing spelling of spam words to avoid detection) to be a major spam content and gave solutions for it. A few teams presented statistical errors such as adding random good words to spam or concatenating of small illegitimate words to form a big permissible word. Teams also discussed about obfuscation of URLs done by encoding or adding unnecessary parenthesis to avoid rule based detection. For data pre-processing different ideas were suggested. Many of them were to filter out stop words like is, an, the and special characters like (), [], performing word stemming and converting all letters into lower case. To counter tokenization attacks strategies suggested included matching of each token against bag of spam words (keyword searching).

Deterministic Finite Automatas were drawn by students for the keywords/spams. Some suggested count of punctuations to be an indicator of spam. Idea to use n-grams approach which takes advantage of contextual phrase information (e.g. “buy now”) was also proposed. For statistical errors different solutions presented were: keeping a count on good words to match against a threshold, weighing the good words against spam words (a significant presence of both can indicate spam) and keeping a count of location of occurrence of good words. For composite attacks, ideas mentioned were use of prefix detection to detect spam by demonstrating the use of REJECT\(^3\) construct of YACC as done in the class. Thus the case helped students to contemplate over different tokenization strategies and gain an experience on how crucial it is to design correct tokenization scheme in the real world design of a spam filter.

C. Case of Human-Robot Chess play (Syntax Analysis)

GOLEMS\(^4\) is a humanoid robotics lab at Georgia Institute of Technology. The lab works towards developing robots having human and even super human capabilities. One of the tasks of the lab is working on building a physical human-robot chess. One side of the chess would have a movable robot arm with sensors providing suitable force to locate, pick, drop and rotate the chess pieces while on other side would be the human playing against the robot. The required objectives of the robot is explained in the case. Developers have come up with controlling of the robot using context-free grammars which they have called as motion grammar. The production rules of the grammar represent a task decomposition of robotic behavior. The motion grammar enables robots to handle uncertainty in the outcomes of control actions through online parsing. The main task is to identify various challenges that will come in design of robot human chessplay system and address those challenges by building the suitable grammar. Thus after understanding the requirements and constraints of the system students are required to suggest a promising motion grammar.

The challenges presented to students in this case are:
1) Identify various requirements of the system to build human-robot chessplay.
2) Identify implicit problems and factors that influence the requirements.
3) Decide and justify the best suitable grammar that can be build which incorporates the requirements of system.

D. Proposed solutions to the case

Each team gave different set of grammars stating different situations they could think can come into human-robot chessplay. While some teams presented a very abstract view of the system in their grammar, few teams did incorporated detailed requirements of the system in their grammar. First the teams identified the tokens in the system. Some worked with taking tokens as chess states (like checkmate, draw), some worked with robot’s movement (like set, release) as tokens while some used sensor’s readings (like pressure release, pressure set) as tokens. Few teams presented the view of making it essential for humans and robot to operate in their own workspace like waiting of the robot’s arm to finish human’s move. Some included the productions of the details of robot’s behavior in handling chess pieces like touching, holding, sliding and reacquiring pieces when they fall. A few did work on providing equal chances to both players and ending the game on either a human win, robot win or a draw. Some of the team focused on working on the mechanics of the robot’s arm like stretching, turning and grasping the piece. Students also worked on the grammar structure to reset the board like making space on the home square if some piece is already occupied. A few of the teams worked on including productions for different chess strategies like for en passant move taking the captured pawn and moving its own pawn to destination square. Some presented semantic actions also with the grammar. Teams also build motion parser using different bottom up parsing techniques discussed in the class. Thus by this case, students gained insights on how to create grammar for a real software.

IV. Evaluation

The experiment is carried on compiler design course offered to 48 third year B.Tech students of Deen Dayal Upadhyaya\(^5\) college, affiliated to Delhi University. We conduct survey adapted from a national survey on faculty perceptions of benefits and challenges of case-based instruction [12] for students to understand their response to case-based approach.

\(^3\)http://dinosaur.compilertools.net/lex/
\(^4\)http://www.golems.org/projects/krang.html
\(^5\)http://dducollegedu.ac.in/
The case discussion increased my interests.

The case study added a lot of realism to

I felt the use of case-based learning was

The case study was helpful in synthesizing

The case study allowed me to view an issue

The case study added a lot of realism to

I was more engaged in class during case

The case discussion increased my interests in

The case discussion strengthen my commu-

The case discussion increased my confidence to

We analyze the survey using frequency distribution and chi square test of association to analyze if students agree that case discussion enhanced different learning principles like learning, critical thinking, engagement, communication skills and team work [13].

Survey questions categorised under different learning principles along with percentage of students choosing the given option is shown in Table I. The frequency of individual survey questions is aggregated to give each of the learning principles- learning, critical thinking, engagement, communication skills and team work a total frequency score and chi-square analysis is done over it. Values of \( \chi^2(2) = 38.7 \) and \( p=0.00013 \) suggests that there is relationship and association between different learning principles. Thus significantly more students agree that case studies increase the skills of learning, critical thinking, engagement, communication and team work.

V. CONCLUSION

We propose and successfully implement the CPLC teaching environment for teaching compiler design concepts using case-based and project-based pedagogy. With case-based pedagogy students gave positive feedback to have learned the course, developed skill of critical thinking about an issue, being actively involved in the course and having improved communication skills and team work. The hands on experience on project gave more practical experience of designing the compiler by themselves. The positive measurement of effectiveness of case-based discussion along with practical experience of compiler designing through projects proves that both of these pedagogy are suitable for teaching concepts of compiler.

REFERENCES


<table>
<thead>
<tr>
<th>Learning Principles</th>
<th>Question</th>
<th>Strongly Agree (%)</th>
<th>Agree (%)</th>
<th>Disagree (%)</th>
<th>Strongly Disagree (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning</td>
<td>I felt the use of case-based learning was relevant in learning about course concepts.</td>
<td>14.5</td>
<td>68.7</td>
<td>10.4</td>
<td>6.25</td>
</tr>
<tr>
<td>Learning</td>
<td>The case-based learning allowed for a deeper understanding of course concepts.</td>
<td>10.4</td>
<td>58.3</td>
<td>27.0</td>
<td>4.1</td>
</tr>
<tr>
<td>Learning</td>
<td>The case study will help me to retain different aspects of compilers better.</td>
<td>12.5</td>
<td>52.0</td>
<td>25.0</td>
<td>10.4</td>
</tr>
<tr>
<td>Critical Thinking</td>
<td>The case study allowed me to view an issue from multiple perspectives.</td>
<td>22.9</td>
<td>64.5</td>
<td>8.3</td>
<td>4.1</td>
</tr>
<tr>
<td>Critical Thinking</td>
<td>The case study was helpful in synthesizing ideas and information presented in course.</td>
<td>12.5</td>
<td>77.0</td>
<td>6.25</td>
<td>4.1</td>
</tr>
<tr>
<td>Critical Thinking</td>
<td>The case study added a lot of realism to class.</td>
<td>14.5</td>
<td>64.5</td>
<td>14.5</td>
<td>6.25</td>
</tr>
<tr>
<td>Engagement</td>
<td>I was more engaged in class during case study.</td>
<td>27.0</td>
<td>62.5</td>
<td>4.1</td>
<td>6.25</td>
</tr>
<tr>
<td>Engagement</td>
<td>The case discussion increased my interests in learning about compilers.</td>
<td>18.75</td>
<td>58.3</td>
<td>14.5</td>
<td>8.3</td>
</tr>
<tr>
<td>Communication Skills</td>
<td>The case discussion strengthen my communication skills to speak in front of audience.</td>
<td>27.0</td>
<td>58.3</td>
<td>12.5</td>
<td>4.1</td>
</tr>
<tr>
<td>Team Work</td>
<td>The case discussion increased my confidence to work in a team.</td>
<td>22.9</td>
<td>62.5</td>
<td>12.5</td>
<td>2.0</td>
</tr>
</tbody>
</table>