Abstract—Skills like logic, reasoning, algorithm formulation for problem-solving, mathematical inferences, application of knowledge and others are independent of language we speak. Using these skills, an individual can solve day to day problems quickly. Art of programming (AOP) is a combinational skill. It is composed of several language-independent skills like logic, reasoning, algorithm design, problem classification and others. However, AOP heavily depends on learners knowledge of the English language because the syntaxes of nearly all important and widely used programming languages were designed using the English language. This introduces a problem before non-English learners (NEL) and learners with English as a second language (ESL) to start coding. In my country India our mother tongue is the Hindi language, and most learners are either NEL or ESL. Due to this, teaching programming becomes challenging. This research is the answer to this challenge. To encounter this problem, we developed a desktop application entitled “Hindi-CNL-Coder”. This application helps naive/beginner NEL or ESL programmers to learn to code using their native Hindi-language, which further helps them to learn to code using traditional programming languages like C, Java, and others. This research presents the complete details associated with the “Hindi-CNL-Coder” version 2 app, along with the technology stack, associated reference architecture and other essential elements used to fabricate the app. This research also presents the usefulness of the proposed application in various learning environments.

Index Terms—Art of Programming, Controlled Natural Language, Desktop Application, English as Second Language, Native Language.

I. INTRODUCTION

The English language serves as a pre-requisite for naive/beginner level NEL and ESL programmers. This happens because most of NEL or ESL programmers have difficulty in understanding the English language and all advance and most widely used programming languages derive their keywords, syntaxes, statements, and other elements from the English language. Moreover, the programming language syntax itself produces complications in understanding the art of coding.

To design an easy AOP learning process for NEL and ESL learners, we have to teach them programming using programming elements designed in their native language rather than in the English language. Moreover, the overall coding elements like language structure, keywords, syntax, and others must be simplified. The essential principles of programming like assigning variable names, declaring functions, creating classes, including header files or importing packages, calling functions using call by reference or call by value, function overloading, function overwriting, default values in parameters, handling arrays, handling conditional statements, handling loops, using jump statements, using structures, object handling, garbage collection functions and others must align with the most widely used or targeted programming languages like C, C++, Java and others.

To achieve these objectives, we engineered a desktop application named “Hindi-CNL-Coder”. This app helps naive/beginner level NEL or ESL Indian coders to code using their mother tongue, which is the Hindi language. Use of this application helps in smooth migration of Hindi coders towards standard programming languages like C, C++, Java, and others. To design programming syntaxes and elements using the Hindi language, we used the concept of controlled natural language (CNL). CNL’s are derived using natural languages (NL) by restricting rules and vocabulary present in NL. So we restricted the use of rules and vocabulary present in the Hindi language in order to obtain HINDI-CNLS. As we know that the Hindi language is not a widely used language in the international market and on the internet so nearly no standard Hindi-CNL engine is present in the market. So instead of building the complete Hindi-CNL engine (which is a costly affair), we fabricated a mechanism to handle the problem. Section IV, V and VI discuss in detail issues concerning CNL’s and Hindi-CNL.

Although the Hindi-CNL engine is the core of the system, still, the overall system requires other architectural artifacts and components. Section VII presents the details concerning the reference architecture of the Hindi-CNL-Coder application. This reference architecture helps software developers to design similar applications for other native languages. The tradeoff between various quality attributes heavily depend on the architecture of the system. Due to this system architecture becomes a critical issue.

Section VIII discusses the complete working of the app. This section also presents the step by step procedure of using this application along with its technical details. On the other hand, section IX lists in detail the usefulness and benefits of the app. Whereas section X point toward some critical issues which end users flagged after using the app. Beta testers of the app also pointed out the same set of issues.

We believe that this application help naive/beginner level NEL or ESL coders to understand better the art of programming. We also believe that this application help in smooth migration of naive/beginner NEL or ESL programmers towards the core programming languages like C, C++, Java, and others. This application also provides a native-language based user-friendly environment for learning the art of programming.

II. RELATED WORK

The researches based on theories given by Bloom [1], Pavlov [2], Bandura [3-6], and Webb [7-10] produces, beautiful results in the
form of classroom-based teaching models and frameworks. These frameworks and models help course instructors and teachers to teach effectively. Teaching style heavily depends on learning strategies adopted by the students. The issues concerning the depth of knowledge, knowledge points and content complexity rigor help educators to design courses and their structure [11-13]. Programming courses are no exception. All teaching styles, trends and pedagogical issues which applies on standard courses apply to the coding courses as well. Learning styles like learning by doing (LBD), learning by imitation (LBI), learning by teaching (LBT), learning by memorization (LBM) and others helps educators to choose teaching strategy [14-15]. Researchers showed that students retain nearly 25% of what they see, 50% of what they see and listed and more than 75% if they learn using LBD approach [19-22]. Several online educators designed their online learning systems using the approaches mentioned above [16-18]. Researchers also proved that block-based programming and algorithm-based programming are suitable for beginners [23-24]. Researchers also showed that cognitive score serves as better criteria for selecting students in an online course concerning their marks and pre-requisites [26]. Researchers also listed several tradeoffs which exist among various types of learning style [25]. All these researchers provide pedagogical background for the development of Hindi-CNLCoder app.

The core engine of this application works on the concept of controlled natural language (CNL). The CNL based concept of micro-publishing [27], knowledge representation and knowledge bases [28], CNL in business [29], use of CNL for Semantics of Business Vocabulary and Business Rules and CNL for inferences [30-31] are several types of research which clarify the concept of CNL and its application. In our case, we need Hindi-CNLCoder engine, but no such engine is available, so we constructed one for this application. The overall architecture design for this application handles the quality attribute (QA) tradeoffs efficiently. The layered approach used for designing several components makes the overall design clear and compact. Researchers proposed several ways of handling quality attributes [32-38]. We considered these approaches while designing our app. Software architects talk about architecture patterns for handling QA’s tradeoffs. We used their concept to handle QA’s related tradeoffs while designing this app.

III. PROBLEMS WITH NEL AND ESL LEARNERS

All NEL and ESL learners suffer from several English language-related issues which include problem in (i) understanding English text/instructions (UETI), (ii) writing English text/instructions (WETI), (iii) filling English templates/forms (FETF), (iv) Interpretation (INTP) and (v) communication (COMM). Apart from these five issues listed above, several issues like (a) interspeaker, (b) intraspeaker, (b) sociolinguistic and (d) others are essential from a linguistic point of view. However, they are not necessary for this research, so we are not going to discuss them here.

The complete set of English language grammatical rules, vocabulary, syntaxes and structures amplify all five problems listed above. The NEL and ESL learners suffer a lot in short duration courses where a good knowledge of the English language serves as an unsaid pre-requisite. Due to English based syntaxes programming courses on C, C++, Java and other standard programming languages make learning difficult for NEL and ESL learners. In practice, the course instructor uses his/her experience to overcome these challenges. Teaching using dual language is the most common approach which they follow. This approach works well for offline teaching mode but fails for teaching online courses. It fails because, in online courses, students from different countries can participate which bars course instructor from using dual or multi-lingual teaching approach. The application proposed in this paper is the answer to this problem.

Most Indian teachers and course instructors follow “just enough English” approach for teaching AOP to NEL and ESL learners. This approach uses limited English keywords, syntaxes and other elements to teach programming to the NEL and ESL students. Instructors and teachers use abstraction for teaching English elements to the learners due to which they highlight only essential details of English elements without going into its core grammar. This abstraction based method seems useful, but in reality, it is a crude method because everything is done informally without any proper control over elements of English. We noticed that programmers produced using this approach lack self-confidence because they mugged up several keywords, syntaxes, concepts and other programming elements without understanding them.

The other approach to handle the above-listed problems is to formally teach the NEL and ESL students a limited set of English language rules and vocabulary, which is domain-specific concerning AOP before teaching them programming. This type of English is known as controlled English, and it falls under the category of CNL. This approach is effective concerning the previous approach. However, researchers observed that this process is time taking and AOP requires limited knowledge of the English language, so it is hard to derive a limited set of English language rules and vocabulary for the same.

In this research, we followed the modified version of the second approach discussed above. The second approach is also known as the controlled natural language (CNL) approach. We modified it, considering the pedagogical issues associated with the AOP.

In order to do a course generally, course instructors choose pre-requisites and cumulative grade point average (CGPA) as screening criteria for students. Researchers showed that cognitive score is more effective screening criteria concerning pre-requisites and CGPA. They further showed that the best way to learn to program is through imitation. We considered these researches while fabricating Hindi-CNLCoder engine. In the next section, we will discuss the basics of CNL and learning by imitation (LBI). These two elements form the conceptual structure for this research.

IV. UNDERSTANDING THE CONTROLLED NATURAL LANGUAGE AND LEARNING BY IMITATION

A controlled natural language (CNL) is a restricted or controlled version of some natural language (NL). A CNL is obtained by extracting a subset of rules and vocabulary from the complete set of rules and vocabulary present in the source natural language. For example, English is a natural language, and several English based CNLs are attemptto, clear talk, ASD simplified English, common logic controlled English, E-prime and others. These English CNL engines are available online.

NLs are used to create CNLs. On the other hand root of NL is embedded in classic literacy which means the art of reading and writing using native NLs. Classic literacy is the primary prerequisite for learning courses like science, arts, technology, social science and others. Learning a new concept or technology requires knowledge of the native NL. Researchers showed that all human brains are designed to think and understand using their native NL. It is to be noted that bilingual or multilingual speakers have two or more languages as their native NLs, respectively.

Initially, CNLs were created to solve five linguistic problems listed in section III. Later they were used in several automated tasks like knowledge representation, formal logic and others. The
translation of a document written in a particular natural language (say NL1) may results in ambiguous interpretations when translated to some other natural language (say NL2) or when read by different persons using NL1 or NL2. This ambiguity arises because NL’s have inbuilt variations like linguistic, interspeaker, intraspeaker, sociolinguistic and others. These variations affect several aspects of language, including phonemes, morphemes, syntactic structures, meanings and others, resulting in ambiguous interpretations. To overcome such ambiguities, CNL’s are used.

Many multinational companies and government agencies started using controlled natural languages for doing their official work. This eliminates interpretation ambiguity while interpreting documents written using CNL’s. CNL’s provide some beneficial properties to users; these properties are as follows:

- CNL’s are easy to learn.
- Sentences written using a CNL are having unambiguous meanings.
- A document written using a CNL has exactly one interpretation.
- Translating a document written using one type of CNL to another type of CNL is easy and unambiguous.
- Native-CNL’s provide links between two or more natural languages.

So from the above discussion, it is easy to conclude that a CNL is the restricted version of NL. When the use of grammatical rules and vocabulary present in any NL is restricted concerning some domain, then the restricted NL obtained is known as controlled natural language.

Researchers showed that our brains are tuned to think, understand and memorize using native natural languages. When we try to learn concepts written in foreign natural languages, our brain translates them to native natural language, which eases the learning process.

Learning programming is always a painful task for beginners. This pain amplifieds when beginners fall under NEL and ESL categories. Is it possible to learn to code using native-controlled natural language? How coding through native controlled natural language help programmers to learn traditional programming languages like Java, C, C++ and others, better? How native-CN help programmers to enhance their coding skills? This research is the answer to these questions.

The best way to learn a natural language is through imitation. The programming language is no exception. So learning by imitation approach is the most suitable approach for learning programming. In one of our paper, we proposed a model which helps naive programmers to learn to program using LBI, learning by doing (LBD) and memorization. This model produces fantastic results when implemented in online coding platform.

The concept of analogy based learning by imitation is used while designing the Hindi-CNLP-coder application. According to the concept of analogy based LBI coding is done using elements of Hindi-CNLP, and these elements are analogous to the elements of standard programming languages like C, C++, Java, Python and others. Due to this migration from coding using the Hindi language towards standard programming languages becomes easy.

V. NO CNL FOR THE HINDI LANGUAGE

Several CNL engines are available online for languages like English, French, Spanish and others but no such engine exists for the Hindi language. After reading several official Hindi documents, we came to know that from several years India is using several Hindi and Urdu language-related official rules, vocabulary and formats for executing the official work. We found that these formats are equivalent to Hindi-CNL and Urdu-CNL, respectively.

Based on these official Hindi-CNL documents, we constructed a template-based Hindi-CNL engine. This engine forms the core of Hindi-CNL-Coder desktop application. In the next section, we will discuss issues concerning the designed engine.

VI. TEMPLATE BASED HINDI-CNL ENGINE AND IT’S COMPARISON WITH C PROGRAMMING LANGUAGE

The Hindi-CNL engine consists of Hindi character set, Hindi keywords, rules for creating identifiers and templates equivalent to standard programming concepts. Let us discuss each of them in detail. The standard Hindi character set consist of 13 SWARS, 36 VYANJANS, 10 –digits 0 to 9 (see figure 1). The Unicode and digital way of writing Hindi character set is shown in table I. The rules for creating identifiers in Hindi-CNL program are as follows:

- The first character of an identifier must either be an underscore or SWAR or VYANJAN.
- Any combination of SWAR and VYANJAN can be used to create an identifier.
- An identifier must not start with digit.
- No special character other than underscore can be used to create identifier.
- Digits can be used to create identifiers but underscore or SWAR or VYANJAN must precede them.
- Total length of the identifier must not exceed 32 characters.
- Keywords cannot be used as identifiers.

The keywords are reserved words in Hindi-CNL. The set of selected (total seventeen words are selected) Hindi words constitutes the keywords. Table II shows the complete list of keywords used in Hindi-CNL.

We used the template-based approach for designing the Hindi-CNL engine because this approach reflects analogy based LBI. In this approach, programming concept-based coding templates are designed using Hindi-CNL. These templates are analogous to their equivalent concepts in standard programming languages like C, C++, Java and others. So learning to code using these templates helps in smooth migration of NEL and ESL students from Hindi-CNLP based coding towards coding using standard coding languages. Tables III-V shows some sample templates and their equivalent C language code.

Table III shows the concept of variable creation in Hindi-CNL and its equivalent code in the C programming language. In this table, some elements required for creating variables are also given like data types in Hindi-CNL and their equivalent in the C language.

Table IV shows the basic Hindi-CNL program template and its equivalent C code. From Table IV, it's clear that the basic structure of the C program consists of header file inclusion, followed by the body of the main program. The analogue of this structure is replicated in Hindi-CNL coding template. The first line of Hindi-CNL code tells the computer that the main program is going to start, and lines second and fourth encapsulates the program body. The third line tells the computer to attach all-important header files in the program.

Table V shows the basic looping template used in the Hindi-CNL, and it's associated equivalent C language code. In the Hindi-CNLP code first line initializes an integer variable second line is the loop syntax and lines 3 and 5 encapsulates the loop body. The loop written in table V prints the first ten integers in reverse order. Line 4 represents the Hindi-CNLP print statement.

Other similar templates form the core of Hindi-CNLP engine.
Now let's have a look at reference architecture of the app.

**TABLE I. SHOWING OFFICIAL UNICODE CONSORTIUM CODE CHART FOR DEVANAGARI SCRIPT.**  
(image source: https://en.wikipedia.org/wiki/Devanagari_(Unicode_block))

<table>
<thead>
<tr>
<th>Line No.</th>
<th>Hindi-CNl Code</th>
<th>Equivalent C Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>शुरुः-लघु</td>
<td>float p;</td>
</tr>
<tr>
<td>2.</td>
<td>अनुभव</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>समताः</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>समष्टि</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>सत्या, सौभाग्य</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE II. KEYWORDS USED IN HINDI-CNl**

<table>
<thead>
<tr>
<th>क्षेत्र-देश-प्रदेश</th>
<th>शुरुः-लघु</th>
<th>अनुभव</th>
</tr>
</thead>
<tbody>
<tr>
<td>समताः</td>
<td>समष्टि</td>
<td>सत्या, सौभाग्य</td>
</tr>
</tbody>
</table>

**TABLE III. SHOWING VARIABLES AND RELATED ELEMENTS IN HINDI-CNl AND THEIR EQUIVALENT CODE IN C PROGRAMMING LANGUAGE.**

<table>
<thead>
<tr>
<th>Hindi-CNl Code</th>
<th>Equivalent C Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>पूर्ण-संख्या</td>
<td>Int</td>
</tr>
<tr>
<td>अनुभव</td>
<td>Float</td>
</tr>
<tr>
<td>समताः</td>
<td>int p;</td>
</tr>
</tbody>
</table>

**TABLE IV. SHOWING BASIC HINDI-CNl PROGRAM TEMPLATE AND ITS EQUIVALENT CODE IN C PROGRAMMING LANGUAGE.**

<table>
<thead>
<tr>
<th>Line No.</th>
<th>Hindi-CNl Code</th>
<th>Equivalent C Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>शुरुः-लघु</td>
<td>float p;</td>
</tr>
<tr>
<td>2.</td>
<td>अनुभव</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>समताः</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>समष्टि</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>सत्या, सौभाग्य</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE V. SHOWING BASIC LOOP CONCEPT USING HINDI-CNl LOOP PROGRAM TEMPLATE AND ITS EQUIVALENT CODE IN C PROGRAMMING LANGUAGE.**

<table>
<thead>
<tr>
<th>Line No.</th>
<th>Hindi-CNl Code</th>
<th>Equivalent C Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>शुरुः-लघु</td>
<td>float p;</td>
</tr>
<tr>
<td>2.</td>
<td>अनुभव</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>समताः</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>समष्टि</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>सत्या, सौभाग्य</td>
<td></td>
</tr>
</tbody>
</table>

**VII. REFERENCE ARCHITECTURE OF HINDI-CNl-CODER APP**

This section presents the complete reference architecture of Hindi-CNl-Coder desktop app. The reference architecture is composed of three engines, namely (i) User Interface engine, (ii) Hindi-CNl engine and (iii) execution engine (see Figure 2). All engines, along with their core elements, are described next.

**A. User Interface Engine**

The User Interface engine allows users to interact with the application. It is composed of a set of Hindi fonts and Hindi text editor for editing programs using Hindi-CNl. It contains Hindi-CNl palette which helps users to do the editing work speedily. This engine also contains a Hindi I/O error dialog. This dialog displays runtime/compile-time errors/exceptions to the user.

**B. Hindi-CNl Engine**

The layered design pattern is used for designing the Hindi-CNl engine. According to this design pattern, the bottom layer provides services to the layer above it. This engine is the core component of this app. Its modeled using analogy base LBI approach and template-based Hindi-CNl approach discussed in the above sections. This engine is composed of three layers and one translator module. These layers, along with the translator module, are discussed next.
1) Hindi I/O Translator Module
   This module helps to translate input text/program to a standard Hindi program format which pre-processor layer can handle. This module translates output generated by the system in user-readable Hindi format because the system generates output in non-Hindi format.

2) Hindi Code Pre-processor
   The Hindi-CNL engine cannot directly parse the Hindi code written by the user. So this layer pre-processes the Hindi code making it compatible for processing by the layer above it. This layer pre-processes the Hindi-CNL templates and Hindi code syntax to make processing smooth.

3) Hindi-CNL-IMC Code Parser
   This layer generates the intermediate Hindi code. This intermediate Hindi code is optimized and free from certain ambiguities which Hindi programmer may unknowingly insert in Hindi code while typing the code.

4) IMC-Hindi-CNL to Java Parser
   This layer parses the intermediate Hindi code generated by Hindi-CNL-IMC Code Parser layer to its equivalent Java code. If during parsing any error occurs then this layer directly contact the “Hindi I/O Error dialog” present in the user interface module and halt the execution till user correct the code and restart the execution. Otherwise, if all is correct, then the equivalent java code is subjected to standard Java compiler and runtime environment.

C. Execution Engine
   The execution engine is designed using the modular design approach. This engine helps in executing the user program. The Hindi-CNL engine generates the Java program equivalent to Hindi source code entered by the user. This Java program is compiled and executed by this engine. This engine contains three main components which are discussed below.

1) Java Compiler and Runtime Engine
   The Java compiler and runtime engine is composed of the standard Java compiler and Java runtime environment (JRE). This engine takes Java program as the input, compiles it and if correct runs it. If any runtime or compile-time error/exception is generated, then it is sent to the Error Mapping Module. If everything is correct, then using the Hindi I/O mapper module and Error mapping module program is executed.

2) Error Mapping Module
   This module helps in handling runtime/compile-time errors/exceptions. This module also uses Hindi I/O mapper to establish proper communication between the execution engine and Hindi-CNL engine. All results generated by Java engine are in English this module along with the Hindi I/O mapper module and Hindi I/O translator module present in Hindi-CNL engine converts them in Hindi-CNL for NEL and ESL users.

3) Hindi I/O Mapper
   This module helps in establishing a communication link between the execution engine and the Hindi-CNL engine. This module also coordinates with other modules like Error mapping module and Hindi I/O translator module for the smooth working of the system.
IX. USEFULNESS, BENEFITS AND APPLICATION OF HINDI-CNL-CODER DESKTOP APPLICATION

We floated an online C programming course from March-2020 to June-2020 for beginner/naive NEL and ESL students. A total of 80 students registered in the course. We used Hindi-CNL-Coder desktop application to teach them coding in Hindi-CNL and then using the concept of analogy based LBI taught them C programming language. We asked them to write algorithms in Hindi for the selected set of programming problems before the course start. This exercise is the pre-test for students. Based on six programming parameters listed later in this section, we scored them. Based on their scores, we clustered them in three groups, namely High performing group (HPG), Moderate performing group (MPG) and Low performing group (LPG). We selected six programming parameters, namely total performance (TP), concept grip (CG), innovation (IN), logic (LO), coding structure understanding (CSU) and problem-solving time (PST) to measure student performance. After the completion of the course, all students were tested and scored on the scale of [0,10] concerning the selected programming parameters. These scores were compared with the scores obtained at pre-test.

We found a remarkable hike in programming parameter for all three groups. Table VI presents the performance hike results concerning selected programming parameters for all three groups.

From Table VI we can easily see that concerning TP, IN and LO highest hike is observed by MPG group which is {25%, 27%, 29%} followed by LPG and MPG groups which is {17%, 20%, 21%} and {13%, 17%, 20%}, respectively.

We also observed that PST is reduced by 24%, 13%, 11% for MPG, LPG and HPG groups, respectively. The CG hike observed is 30%, 28%, 20% concerning HPG, MPG and LPG respectively. The CSU shows 18%, 33%, 18% hike concerning HPG, MPG and LPG groups.

Apart from this, we asked students to comment concerning the Hindi-CNL-Coder desktop app. Most students gave powerful positive comments. They claimed that this app could enhance their programming capability and programming skills effectively. They loved the concept of analogy based LBI. They also asked to design similar app for other native languages of India.

Few students claimed that this way of migrating from Hindi-CNL code towards the C language code is easy and enjoyable. They also personally told us that they could understand linking concepts easily just by using this application. Many students asked us to configure the application for Java and Python. They suggested us to launch the online version of this application. They also suggested us to find new editing methods for native Indian languages.

Intelligent Tutoring Systems (ITS) and Massive Open Online Courses (MOOC) help online educators for teaching programming language courses. They follow learning by doing approach to teach programming. As these systems help in teaching standard programming languages, so learning problems mentioned above persists for NEL and ESL learners. The designed application can be tuned concerning learners native languages to teach them the core concept of programming. Then they can easily be migrated towards the standard languages. So online educators can use tuned versions of our application and integrate it with their online education system. This integration allows NEL and ESL learners to use our app as prerequisite as well as concept understanding aid for learning programming.

The proposed application is light weighted, and its mobile version can change the learning/coding trend for online learners. We can also model block-based or flowchart-based programming paradigms in this app to make learning programming fun. Such integration of different programming paradigms helps in better visualization of programming concepts to the NEL and ESL learners.

TABLE VI

<table>
<thead>
<tr>
<th>PROP</th>
<th>CLUST</th>
<th>TP</th>
<th>CG</th>
<th>IN</th>
<th>LO</th>
<th>CSU</th>
<th>PST</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HPG</td>
<td>13</td>
<td>30</td>
<td>17</td>
<td>20</td>
<td>18</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>MPG</td>
<td>25</td>
<td>28</td>
<td>27</td>
<td>29</td>
<td>33</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>LPG</td>
<td>17</td>
<td>20</td>
<td>20</td>
<td>21</td>
<td>18</td>
<td>13</td>
</tr>
</tbody>
</table>

Where: CLUST=Cluster; PROP=Programming Parameters; TP=Total Performance; CG=Concept Grip; IN=Innovation; LO=Logic; CSU=Coding Structure Understanding; PST=Problem Solving Time; HPG=High Performance Group; MPG=Moderate Performance Group; LPG=Low Performance Group.

X. RESEARCH LIMITATIONS

Although the application is very successful for naive/beginner NEL and ESL students, still it suffers from two significant problems. The first problem is the Hindi text editing problem. In India, English keyboards are used, so for NEL and ESL students typing become a problematic issue even if we make keys with Hindi alphabets. In reality, Hindi typing is problematic. All students want us to design better Hindi typing or input mechanism. The Hindi-CNL palette provided with the app helps students significantly. However, still, we have to find new ways for Hindi typing. The second problem is with the errors produced by the compiling program. Since all errors/exceptions thrown by Java engine are in English and we only mapped a few important errors/exceptions in Hindi, so in case of non-mapped errors/exceptions system display them in English. To solve this issue, we have to design a complete error mapping system.
XI. Conclusion

This research focuses on architectural issues concerning the Hindi-CNLCoder desktop application. The paper started by discussing the problems that NEL and ESL users face while learning to code. Then it discusses issues like program syntax, structure, keywords, rules and others, which make coding a frustrating task for beginner level NEL and ESL learners. Then concepts concerning controlled natural languages and their use to teach programming courses to beginner level NEL and ESL students were discussed. With proper references, we presented essential properties concerning CNL’s and their application in teaching foreign languages unambiguously. We discussed the use of CNL properties for teaching courses concerning programming.

Knowing the fact that one can teach English-CNLC to students using the Hindi-CNLC, the research extended the concept of Hindi-CNLC to learn to code in the Hindi language followed by migration step towards the standard programming languages like C, C++, Java, Python and others. The complete learning approach is termed as analogy based LBI.

This research also discusses the design-related technical issues concerning the Hindi-CNLC engine, because no standard Hindi-CNLC engine is present in the market. So this paper presents in detail the fabrication process of the template-based Hindi-CNLC engine.

Then this research focuses on the reference architecture of the app. This architecture helps software designers to design similar applications for other native languages. This paper further showed the internal components of the app, namely Hindi-CNLC engine, Execution engine and User interface engine while presenting the reference architecture.

Further, this paper explains the complete working of the app in detail. In working the complete flow of control is explained with the help of a diagram. Then the paper discusses usefulness and benefits related to Hindi-CNLC-Coder desktop application. To highlight the benefits we presented the performance hike results for 80 students who used this application for learning coding in C programming language. We also listed their feedbacks to show usefulness and benefits of this application. Then finally, we listed two significant issues faced by students while using this application. The suggestions given by students concerning the issues present in the app will be considered in the next design iteration of the application.

We strongly believe that the development of such systems and applications helps NEL and ESL beginner level students to learn to code productively.

REFERENCES
program, he was working as Lecturer at Central Institute of Plastics Engineering and Technology, Lucknow, India. He is having more than ten years of teaching experience.

Umesh Chandra Jaiswal is working as Professor in the department of computer science and engineering, Madan Mohan Malaviya University of Technology, Gorakhpur, India. He is interested in the field of Natural Language Processing, Algorithms and Machine Learning. He can be contacted at ucj_jaiswal@yahoo.com.

Shalini Lamba is Head of Department of Computer Science in National PG College Lucknow, India. She is also the Placement Coordinator of her college. She is an active member in the Board of Studies and Academic Council. She has interests in the field of Data Mining, Data Analytics besides software development. She has more than twenty years of teaching experience. You may reach her at drshalinilamba@gmail.com/placement@npge.in.

Prabhakar T.V. is interested in building system and still feels excited about science and technology. He worked in multiple areas like Databases, Logic Programming, User Interface Design, Internet Technologies, Software Architecture, Knowledge Modeling, Indian Language Technologies. He worked with CMC R&D, Satyam Computer Services and have been a consultant to a large number of IT organizations and e-governance initiatives. He is currently working as Professor in the department of computer science and engineering, Indian Institute of Technology, Kanpur, India. He can be contacted at tvp@iitk.ac.in

Siddharth Srivastava is a PhD research scholar in the department of computer science and engineering at Indian Institute of Technology, Kanpur, India. He is currently working in the field of software architecture. He loves to develop applications for improving Indian education sector using software architecture and artificial intelligence. Before joining the PhD program, he was working as Lecturer at Central Institute of Plastics Engineering and Technology, Lucknow, India. He is having more than ten years of teaching experience.


