PSEUDOCODE TO SOURCE CODE TRANSLATION

[1]Vishal Parekh, [2] Dwivedi Nilesh
[1]B.E. Student, [2]B.E. Student
[1,2]Department of Computer Engineering,
[1,2]L. D. College of Engineering, Ahmedabad, India

***Abstract*— In the field of software engineering most programming languages have certain idiosyncrasies. Due to this, beginner programmers or programmers who are learning a new programming language have difficulty in writing good and efficient programs. We are creating a translation program that automatically converts a well specified pseudo code to source code in a particular programming language like C/C++/JAVA using neural network techniques. This program allows the user to specify his/her code in more or less immutable natural language and it removes current gap between programming language and natural language. Our program allows the user to focus on logic to convert input to output rather than focusing on programming language implementation.**

***Index Terms*— Pseudo code, Mapper, Translator, Programming Languages, Neural Network, XML Tags**

I. INTRODUCTION

Software has been applied in diverse areas like Microbiology, Astronomy, Social Sciences and many other areas of modern life. Today, everything seems to be digital, driven by software. Advances in software are powering rapid changes in technology, impacting us as individuals and our society. In recently years, the applications of software have dominated in many fields of our life. First, there is office software such as Microsoft Office and Open Office which brought us a huge improvement in the efficiency of study and work related to the text editor. Secondly, with the wide application of communication software such as Skype, MSN, Yahoo Message, the communication between people become more and more convenient. [17]

The person has experienced the power of the software as more and more problems are lending themselves to algorithmic solutions. And all the software's are the implementation of an Algorithm written in a programming language. Algorithms, however, need to be implemented; and this is where a problem lies. Veteran Programmers can easily implement an algorithm and write good and efficient programs. In contrast, this process is much more difficult for beginner programmers or programmers who are learning a new programming language. Such inexperienced people sometimes do not understand the paradigm and style of the programming language at hand, so implementing an algorithm in such languages imposes a large burden. Moreover, there are many programming languages which allow coding in a variety of paradigms. So it is not easy for someone not trained in Computer Science to convert a program from one language to another. [3]

We propose a translation process that allows the people to write the algorithmic solutions of the problem in the form of Pseudocode which can further be translated into a programming language like C, C++, JAVA, etc.

II. THE CONVERSION PROCESS

The conversion process is a step-by-step process, there are several processes involved in this. For a better conversion process, the system should not be just a simple syntax replacement system. It should also preserve the structure or should modify it to make it even better by removing redundant codes. Hence, it will be better if a translator is designed for this purpose. And the purpose of translation is to convert the given pseudocode into its corresponding XML Specification file. This XML Specification file can be converted into any programming language using mapper.

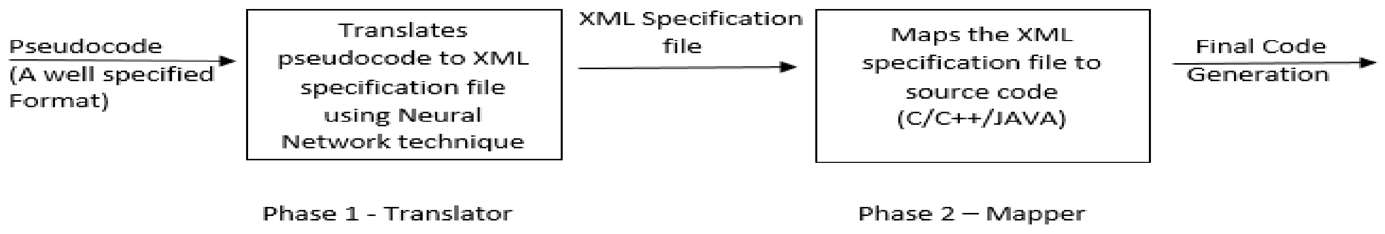
Pseudocode
(A well specified Format)
Translates pseudocode to XML specification file using Neural Network technique
XML Specification file
Maps the XML specification file to source code (C/C++/JAVA)
Final Code Generation
Phase 1 - Translator
Phase 2 – Mapper

Figure –1 the Conversion Process

The entire process of translating a pseudocode to any programming language will be divided into two phases:

Phase 1 (Translation phase): Translating the pseudocode to XML specification file using cascade back propagation neural network.

Phase 2 (Mapping phase): Mapping the XML specification file of the pseudocode with the programming language syntax and generating the final code.

In order to carry out the two phases we will require building a translator using cascade back propagation neural network which will be independent of the programming language and a mapper which will be dependent on the programming language.

For example: For conversion of pseudo code to C,C++ or JAVA, The translation phase (pseudocode to XML) remain same for all three but the mapping phase (XML to source code) will be different for each of the programming languages.

Let us discuss the two integral part of this process in detail:

A. Translator

There are several neural network that can be used to convert pseudo code to XML specification file. In the paper "Automatic Pseudocode to Source Code Translation Using Neural Network Technique", [2] the author concluded the best type of neural network for converting pseudo code to program is a cascade back propagation neural network depending on the training and testing time.

Input to the translator:

Pseudocode is an artificial and informal language that helps programmers develop algorithms. Pseudocode is a "text-based" detail (algorithmic) design tool.
The rules of Pseudocode are reasonably straightforward. Examples below will illustrate this notion.

Example:

```
integer student_grade
read student_grade
if student_grade >= 35
      print "passed"
else
     print "failed"
```

Translation process:

The entire pseudocode will be read line by line till end. Each line will represent a single operation for the computer to perform. Further each line will be divided into the streams of tokens. For each token there will be an equivalent XML tag in training pairs of neural network. So, the tokens will be replaced by an equivalent XML tag. The process will get repeated until each line is processed.

Output of the Translator:

The translator will return a XML specification file of the pseudocode which will contain the XML representation of the pseudocode. As this stage is programming language independent, so same XML specification file will be used for different programming language.

B. Mapper

The mapping process is all about using regular expressions and matching patterns. A regular expression is a compact way of describing a pattern. The parsing of XML specification file will be carried out.

Input to the mapper:

The XML specification file which contains the XML tag will be the input to the mapper.

For Example:

```
<start>
      <main>
            <var>
                  <type>string</type>
                  <name>a</name>
                  <init>Hello World</init>
            </var>
            <write>a</write>
      </main>
</start>
```

Mapping process:

Mappers will map the XML tags with the operations and represent it according to the syntax of that programming language
For example, following XML tag will be interpreted differently in different programming language according to their syntax and rules:
XML:

```
<write>Hello World</write>
```

C:

```
printf("Hello World");
```

C++:

```
cout<<"Hello World";
```

JAVA:

```
System.out.println("Hello World");
```

Output of the mapper:

After mapping of all XML tags take place, a source file will be generated in the required programming language.

III. DEMONSTRATION

In this section, we put our conversion process from pseudocode to source code to test. We take up an algorithm, write the pseudo code conforming to our formal framework, and then take a look at how the translation process performs.
We will describe the whole process of conversion for the following problem step by step.
Problem Statement: Program to Check Whether a Number is Prime or Not
Input: An integer greater than 1.
Output: "YES" if the number is prime else "NO".
Step 1: Firstly, we will write a pseudocode for the problem, which will be given as an input to the translator.

Pseudocode:

```
1.  integer n
2.  integer i
3.  integer flag = 0
4.  read n
5.  do for i = 2 to n-1
6.        if n % i == 0
7.              flag = 1;
8.        endif
9.  endfor
10. if flag == 0
11.       print "YES"
12. else
13.       print "NO"
14. endif
```

Step 2: Now this pseudocode will be scanned line by line and divided into the stream of tokens. By applying Cascade Back Propagation Neural Network to the pseudo code, we will convert the pseudocode to XML specification file. In this process of translation, the Cascade back propagation neural network will find an equivalent xml tag for each token from the dictionary.

XML Specification file:

```
<start>
        <main>
                <var>
                        <type>int</type>
                        <name>n</name>
                </var>
                <var>
                        <type>int</type>
                        <name>i</name>
                </var>
                <var>
                        <type>int</type>
                        <name>flag</name>
                        <init>0</init>
                </var>
                <vread>n</vread>
                <for>
                        <init>
                                <assign>
                                        <op1>i</op1>
                                        <op2>2</op2>
                                </assign>
                        </init>
                        <terminate>
                                <cond>
                                        <op1>i</op1>
                                        <op2>n</op2>
                                        <op>><</op>
                                </cond>
                        </terminate>
                        <incr>
                                <assign>
                                        <op1>i</op1>
                                        <op2>
                                                <exp>
                                                        <op1>i</op1>
                                                        <op2>1</op2>
                                                        <op>+</op>
                                                <exp>
                                        </op2>
                                </assign>
                        </incr>
                        <exec>
                                <sif>
                                        <if>
                                                <cond>
```

```
                            <op1>
                                <exp>
                                    <op1>n</op1>
                                    <op2>i</op2>
                                    <op>%</op>
                                <exp>
                            </op1>
                            <op2>0</op2>
                            <op>=</op>
                        </cond>
                        <exec>
                            <assign>
                                <op1>flag</op1>
                                <op2>1</op2>
                            </assign>
                        </exec>
                    </if>
                </sif>
            </exec>
        </for>
        <sif>
            <if>
                <cond>
                    <op1>flag</op1>
                    <op2>0</op2>
                    <op>=</op>
                </cond>
                <exec>
                    <write>YES</write>
                </exec>
            </if>
            <else>
                <exec>
                    <write>NO</write>
                </exec>
            </else>
        </sif>
    </main>
</start>
```

Step 3: The step 1 and step 2 are programming language independent. Step 1 and Step 2 will remain same irrespective of the programming language of our final code. In step 3, we will need programming language specific mapper which will map the XML specification file to a particular programming language.

Mapping the XML Specification file to C language:

```c
#include <stdio.h>
int main()
{
    int n,i,flag=0;
    scanf("%d",&n);
    for(i=2;i<n;i++)
    {
        if(n%i==0)
        {
            flag=1;
        }
    }
    if(flag==0)
    {
        printf("YES\n");
    }
    else
    {
        printf("NO\n");
    }
    return 0;
}
```

Mapping the XML Specification file to C++ language:

```
#include <iostream>
using namespace std;
int main()
{
    int n,i,flag=0;
    cin>>n;
    for(i=2;i<n;i++)
    {
        if(n%i==0)
        {
            flag=1;
        }
    }
    if(flag==0)
    {
        cout<<"YES"<<endl;
    }
    else
    {
        cout<<"NO"<<endl;
    }
    return 0;
}
```

Mapping the XML Specification file to JAVA language:

```
import java.util.*;
public class Main
{
    static Scanner sc = new Scanner (System.in);
    public static void main(String a[])
    {
        int n,i,flag=0;
        n = sc.nextInt();
        for(i=2;i<n;i++)
        {
            if(n%i==0)
            {
                flag=1;
            }
        }
        if(flag==0)
        {
            System.out.println("YES”);
        }
        else
        {
            System.out.println("NO”);
        }
    }
}
```

IV. CONCLUSION

In this paper, we proposed a method where the user presents a pseudocode as an input, and the output is an implementation of the pseudocode in a specific programming language. Our method is divided into two phase, 1. Translation Process, which includes Cascade back propagation neural network and 2. Mapping Process, this parses the XML specification file and maps it according to the syntax of programming language. We took up an example to check whether the given number is prime or not, wrote the pseudo code, and then converted to C Language, C++ Language and JAVA Language. Experiments showed that our proposed methods generate grammatically correct source code for the C Language, C++ Language and JAVA Language.

REFERENCES

[1] Mukherjee Suvam and Chakrabarti Tamal, “Automatic Algorithm Specification To Source Code Translation Mukherjee”, Indian Journal Of Computer Science And Engineering (Ijcse), Vol. 2 No. 2 Apr-May 2011.

[2] Dr. safwan Omer Hasson and Fatima Mohammed Rafie Younis “Automatic Pseudocode to Source Code Translation Using Neural Network Technique”, International Journal of Engineering and Innovative Technology (IJEIT) Volume 3, Issue 11, May 2014.

[3] Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura, “Learning to Generate Pseudo-code from Source Code using Statistical Machine Translation”, Conference: 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)

[4] Lili Qiu, “Programming Language Translation”, Department of Computer Science, Cornell University.

[5] Dony George, Priyanka Girase, Mahesh Gupta, Prachi Gupta and Aakanksha Sharma, "Programming Language Inter-conversion", 2010 International Journal of Computer Applications (0975 - 8887) Volume 1 – No. 20.

[6] Karatrantou Anthi and Panagiotakopoulos Chris, “Algorithm, Pseudo-Code and Lego Mind storms Programming”, Proceedings of International Conference on Simulation and Programming for Autonomous Robots/Teaching with Robotics: Didactic Approaches and Experiences, 2008.

[7] AL-Allaf Omaima NA and AbdAlKader Shahlla A, “NONLINEAR AUTOREGRESSIVE NEURAL NETWORK FOR ESTIMATION SOIL TEMPERATURE: A COMPARISON OF DIFFERENT OPTIMIZATION NEURAL NETWORK ALGORITHMS”, UbiCC Journal, [9] Navarro G, "A Guided Tour to Approximate String Matching", ACM Computing Surveys, 33(1):31-88, March 2001.

[8] The Language Guide, University of Michigan, Dearborn Computer Science and Information Science Department. http://www.engin.umd.umich.edu/CIS/course.des/cis400/index.html

[9] XML Tutorial- XML Tree at w3schools; http://www.w3schools.com/xml/xml_tree.asp

[10] XML Tutorial- XML DTD at w3schools: http://www.w3schools.com/xml/xml_dtd.asp

[11] J. M. Boyle and M. N. Muralidharan, "Program Reusability through Program Transformation". IEEE Transactions on Software Engineering 10, 5 (Sept. 1984).

[12] T. R. Kennedy, III, "Using Program Transformations to Improve Program Translation".

[13] M. Si and T. Reps, "Program Generalization for Software Reuse: From C to C++". In Proc.

[14] SIGSOFT '96, 1996.

[15] N. Kushman and R.Barzilay, “Using semantic unification to generate regular expressions from natural language,” in Proc. NAACL, 2013.

[16] T. Lei, F. Long, R. Barzilay, and M. Rinard, “From natural language specifications to program input parsers,” in Proc. ACL, 2013. [8] Olsen Anne L, “Using pseudo code to teach problem solving”, Journal of Computing Sciences in Colleges, December 2005.

[17] Hao Chen, "Comparative Study of C, C++, C# and Java Programming Languages", VAASAN AMMATTIKORKEAKOULU UNIVERSITY OF APPLIED SCIENCES, Degree Program of Information Technology