TRANSLATION OF INDONESIAN NATURAL LANGUAGE TO SOURCE CODE IN PASCAL LANGUAGE

Mohammad Kohar[1], Ken Kinanti Purnamasari[2]

[1,2] Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur 112-114 Bandung 40132
Email : kohar@email.unikom.ac.id[1], ken.kinanti@email.unikom.ac.id[2]

ABSTRACT

Source code in computer science is a collection of commands to solve a problem written in a programming language. But programming languages are usually difficult to understand, because the structure of the language is rigid and unnatural. One way that can be done so that programmers do not need to understand the structure of a programming language is by translating from natural languages to programming languages. Therefore, this study will translate from natural language in Indonesian to source code in Pascal language. The method used in this research is rule-based method. The process on the system has three main stages, namely preprocessing (case folding and filtering), analysis (scanning and parsing), and translation (code generation). The preprocessing stage is done so that the input text is clean of characters that are not needed, then the analysis phase is the process of ensuring the input text is in accordance with the written rules, then enters the translation phase from Indonesian to Pascal language. This study can translate the sequence command which includes creating variables, calling readln, writeln, and basic arithmetic operations. Based on the results of testing of 100 Indonesian command texts containing 8 combinations of commands it produces an accuracy value of around 98%. Indonesian grammar and Pascal language need to be adjusted again in order to handle more complex commands.

Keyword : Translation, Source Code, Pascal Language, Natural Language Processing, Programming Languages.

1. PRELIMINARY

Source code in the computer science is a collection of commands to solve problems or algorithms written in languages that are understood by computers or commonly called programming languages [1]. Source code has writing rules that have been determined by the particular programming language used. When the programmer will make a statement or order to do a process, it must follow the writing rules that are determined by a programming language. However, programming languages are difficult to understand, because the writing structure unnatural. One way to be able to create programs without having to understand the structure of programming languages is by translating from natural languages to programming languages.

Several studies have been conducted to translate from English to source code in *Python* [2] and C [3], and from Indonesian to source code in C ++ [4]. Research conducted by Satu and Avinash [2] already able to translate from natural language in English to source code. In this study [2] can handle collision cases such as entering values, addition, and displaying values. The study [2] did not carry out accuracy testing, so the value of accuracy obtained was not yet known. Then in the research carried out by Nadkarni, Panchmatia, Karwa, and Kurhade [3] and research conducted by Dirgahayu, Huda, Zukhri, and Ratnasari [4], can handle cases of collections, branching, and repetition. It's just that in the study [3] [4] still using input text in pseudocode in English [3] and Indonesian [4]. Until now, researchers have not found research in the case of translation of natural language in Indonesian into the source code.

Based on the description above, this research will build a translator system from natural language in Indonesian to source code in Pascal language. In the translation process, the rule-based method is determined based on the case raised.

2. THEORETICAL BASIS

The cornerstone of theory is the theories that became the reference in this study.

2.1. Natural Language Processing

Natural language processing is one branch of artificial intelligence. Natural language processing examines communication between humans and computers with natural language intermediaries that humans have. In doing so, the natural language sent to the computer must be processed in advance so that it is understood by the computer [5]. One challenge in natural language is the choice of the meaning of a word that has more than one meaning, for example the word 'bisa' in Indonesian, the word 'bisa' can mean 'poison' and can also mean 'can' according to the sentence [6]. Research that can handle cases of words that have more than one

meaning is Part of Speech Tagger, as was done by Purnamasari and Suwardi [7]. Some areas of research in the field of natural language processing are question answering systems, summarization, speech recognition, document classification, and machine translation [5]. Machine translation is a research that focuses on computers that understand human natural language and translate it into other languages [5].

2.2. *Pascal* Language

Pascal language is a programming language developed by Niklaus Wirth around 1970 [8]. Pascal's name was taken from a mathematician named Blaise Pascal. This Pascal language was originally developed for teaching programming languages.

2.3. Scanning

Scanning is the stage of solving input text into tokens based on its class [9]. Then the scanning tokens will be input data at the Parsing stage.

2.4. Parsing

Parsing is a syntactic analysis stage by checking the order in which strings or tokens appear based on predetermined grammar. Grammar itself is a collection of non-terminal symbols, terminal symbols, and initial symbols that are limited by the rules of production [9]. Simply put, parsing is the process of checking input text whether it is in accordance with the rules of the language used or not.

3. RESEARCH METHODS

In this study, using descriptive research methods. Because in this study the facts and characteristics of objects are described systematically [10]. The flow of research in this study includes five stages, namely problem identification, data collection, method analysis, software development, and conclusion drawing. The flow of research can be seen in Figure 1.

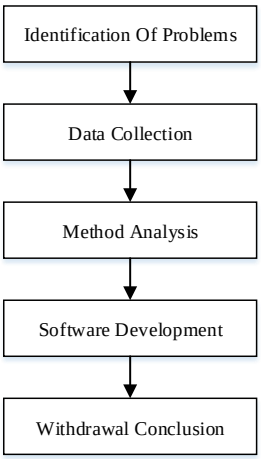

Figure 1 Research Flow

The explanation of the research flow in Figure 1 is as follows.

a. Identification Of Problems
   The problem identification stage is a process of observation of research that has been done before to determine the needs and objectives of the system to be achieved.
b. Data Collection
   In this study, data collection carried out was a literator study both printed and electronic.
c. Method Analysis
   At the stage of method analysis, the methods used in this study will be analyzed, starting from the preprocessing stage, the analysis process, and the translation process.
d. Software Development
   In this study, the development of the software used is the waterfall model. The waterfall model is a method of building software that is sequential in each process [11] [12]. The flow of the waterfall model can be seen in Figure 2.

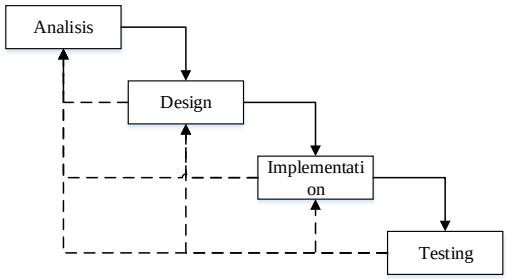

Figure 2 Model *Waterfall* [12]

e. Withdrawal Conclusion
   At the stage of drawing conclusions is an explanation of the results of the research that has been done.

4. RESULTS AND DISCUSSION

This chapter will discuss the research conducted and the results obtained from this study.

4.1. Problem analysis

Source code is a collection of commands to solve a problem written according to the programming language used. Source code format of writing source code is not natural so it is difficult to learn the structure. Therefore we need a system that can translate natural language in Indonesian into source code in *Pascal* language. Here are examples of translations from Indonesian into *Pascal*.

Table 1 Examples of Translation Results

Indonesian	***Pascal***

Buat program penjumlahan. Kemudian buat variabel a dengan tipe data integer.	program penjumlahan;
	var
	  a : integer;
	begin
	  a := 15 + 2;

Tambahkan 15 dengan 2 masukan ke a. Tampilkan nilai a.	writeln(a); end.
Buat aplikasi hitungluas. Buat variabel sisi dan luas dengan tipe data integer. Baca nilai sisi. Kalikan sisi dengan sisi kemudian masukkan hasilnya ke luas. Tampilkan nilai luas.	program hitungluas ; var sisi , luas : integer ; begin readln (sisi) ; luas := sisi * sisi ; writeln (luas) ; end.

4.2. Input Data Analysis

In this study the input data is in the form of an Indonesian text containing sentences that are consecutive to solve a problem. An example of input data in this study can be seen in Table 2.

Table 2 Sample Input Text

No.	Input Text
1	Buat aplikasi inputoutput. Buat variabel x dengan tipe data integer. Baca nilai x. Kemudian tampilkan nilai x.
2	Buat aplikasi hitungvolume. Kemudian buat variabel panjang, lebar, tinggi, dan volume dengan tipe data integer. Baca nilai panjang, lebar, tinggi. Kemudian panjang dikali lebar dikali tinggi masukkan hasilnya ke volume. Kemudian tampilkan nilai volume!.
3	Buat aplikasi hitungluas. Buat variabel sisi dan luas dengan tipe data integer. Baca nilai sisi. Kalikan sisi dengan sisi kemudian masukkan hasilnya ke luas. Tampilkan nilai luas.

4.3. System Overview

In this study, the system that was built will be able to translate Indonesian into the code in Pascal language. The system built has three main processes, namely preprocessing, the analysis process, and the last is the translation process. In preprocessing there are stages of case folding and filtering, the analysis process has stages of scanning and parsing, and the translation process has a code generation stage.

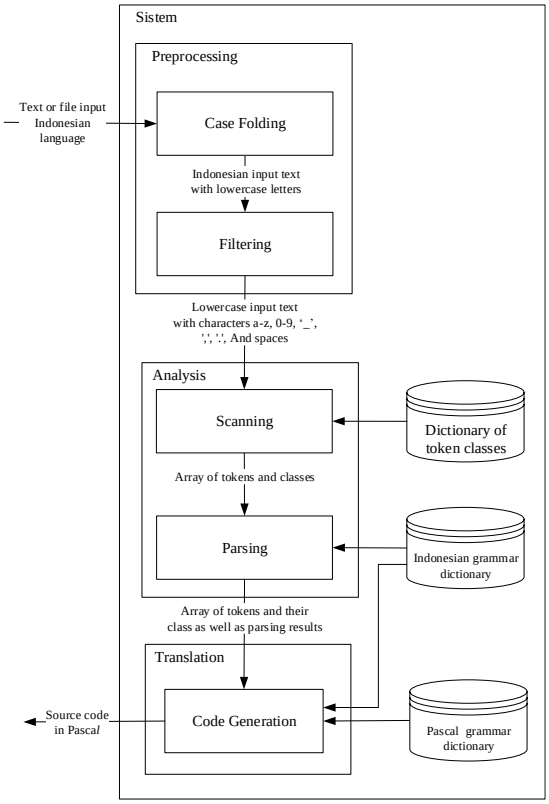

Figure 1 Overall System Block Diagram

The process will be run when the user enters the text of the program creation command in Indonesian.

Table 3 Sample Input Text

Input Text
Buat aplikasi tampil_string. Tampilkan hallo world!.

4.4. Preprocessing

Preprocessing is the first step to prepare Indonesian input texts to be ready for use in the analysis process. Preprocessing has two stages, namely folding cases and filtering.

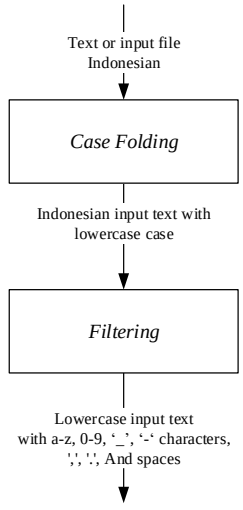

Figure 2 Preprocessing Block Diagram

a. Case Folding
In this study folding cases are used to homogenize letters into lower cases or lowercase letters to facilitate the analysis process.

Table 4 Example of Folding Case Results

Before
Buat aplikasi tampil_string. **T**ampilkan hallo world!.
After
buat aplikasi tampil_string. **t**ampilkan hallo world!.

b. Filtering
The filering stage is the character sorting phase which is considered necessary in this study. Unnecessary characters will be removed or replaced. In this study the characters allowed to enter the translation phase are the characters a-z, 0-9, '_', '-', commas (','), periods ('.'), and spaces.

Tabel 5 Example of Filtering Results

Before
buat aplikasi tampil_string. tampilkan hallo world**!**.
After
buat aplikasi tampil_string. tampilkan hallo world.

4.5. Analysis Process

The analysis process is the stage of analyzing input data whether or not in accordance with the rules. The analysis process has two stages, namely scanning and parsing.

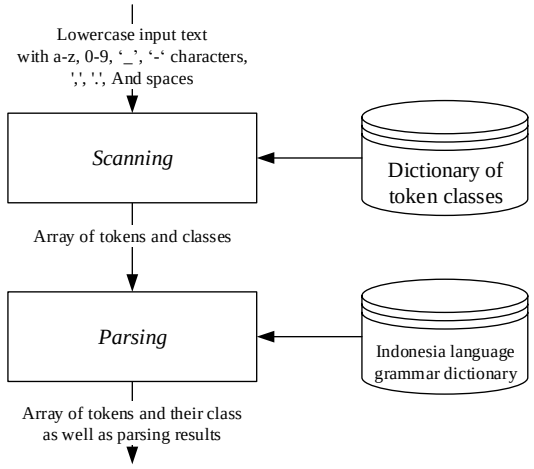

Figure 3 Analysis Process Block Diagram

a. Scanning
Scanning is the stage to change the input text of the preprocessing results into their tokens and classes. Then the tokens from scanning will be used at the parsing and translation process.

Table 6 List of Tokens and Classes

Token	Token Class
'tambah', 'ditambah', 'tambahkan', 'ditambahkan', 'kurang', 'dikurang', 'kurangkan', 'dikurangkan', 'kurangi', 'dikurangi', 'kali', 'dikali', 'kalikan', 'dikalikan', 'bagi', 'dibagi', 'bagikan', 'dibagikan'.	Arithmetic Operator
'program', 'aplikasi', 'variabel', 'peubah', 'tipe', 'tipenya', 'integer', 'string', 'bulat', 'pecahan', 'masuk', 'masukkan', 'dimasukkan', 'isi', 'isikan', 'diisi', 'diisikan', 'tampil', 'tampilkan', 'baca'.	Keyword
'buat', 'buatkan', 'buatlah', 'kemudian', 'lalu', 'dan', 'dengan', 'ke', 'data', 'datanya', 'nilai', 'nilainya', 'hasil', 'hasilnya', 'bilangan'.	Additional Token
[a..z, 0..9, '_']	IdentApp
[a..z, 0..9, '_']	IdentVar
[0..9, ',' , '-']	Number
[a..z, 0..9, '_']	String
',', '.'	Delimiter

The example of scanning results from filtering data in Table 5 can be seen in Table 7.

Table 7 Example of Scanning Results

Before	
buat aplikasi tampil_string. tampilkan hallo world.	
After	
Token	**Class**
buat	AdditionalToken
aplikasi	Keyword
tampil_string	IdentApp
.	Delimiter
tampilkan	Keyword
hallo world	String
.	Delimiter

b. Parsing
The parsing stage is the stage of checking the order in which the Indonesian input text appears as a result of the scanning process. The purpose of the parsing process is to ensure the order in which the tokens appear according to the rules that have been made, so that if there is an input text that is not in accordance with the rules or grammar the translation process will not be executed. In the case example in this study the parsing status was accepted, because all tokens were successfully derived according to

grammar, a decrease in tokens can be seen in Table 8.

Table 8 Contoh Proses Penurunan Token

<START> → <PROGRAM_DECL <DELIMITER BLOCK>
<COMMAND_WORD> <PROGRAM_KEYWORD> <PROGRAM_IDENT> <DELIMITER> <BLOCK>
buat <PROGRAM_KEYWORD> <PROGRAM_IDENT> <DELIMITER> <BLOCK>
buat aplikasi <PROGRAM_IDENT> <DELIMITER> <BLOCK>
buat aplikasi tampil_string <DELIMITER> <BLOCK>
buat aplikasi tampil_string . <BLOCK>
buat aplikasi tampil_string . <STATEMENT>_<BLOCK>
buat aplikasi tampil_string . <STATEMENT_SQUENCE> <DELIMITER>
buat aplikasi tampil_string . <STATEMENT> <DELIMITER>
buat aplikasi tampil_string . <IO_STATEMENT> <DELIMITER>
buat aplikasi tampil_string . <OUTPUT_STATEMENT> <DELIMITER>
buat aplikasi tampil_string . <KEYWORD_OUTPUT> <EKSPRESI_SEQUENCE> <DELIMITER>
buat aplikasi tampil_string . tampilkan <EKSPRESI_SEQUENCE> <DELIMITER>
buat aplikasi tampil_string . tampilkan <EKSPRESI> <DELIMITER>
buat aplikasi tampil_string . tampilkan <EKSPRESI_1> <DELIMITER>
buat aplikasi tampil_string . tampilkan <FACTOR_SQUENCE> <DELIMITER>
buat aplikasi tampil_string . tampilkan <FACTOR> <DELIMITER>
buat aplikasi tampil_string . tampilkan <STRING> <DELIMITER>
buat aplikasi tampil_string . tampilkan hallo world <DELIMITER>
buat aplikasi tampil_string . tampilkan hallo world .

4.6. Translation Process

The translation process is the process of translating Indonesian text into Pascal's language source code. This translation process will be carried out when the status is parsed at the analysis stage 'received'. The translation process has one stage, namely code generation.

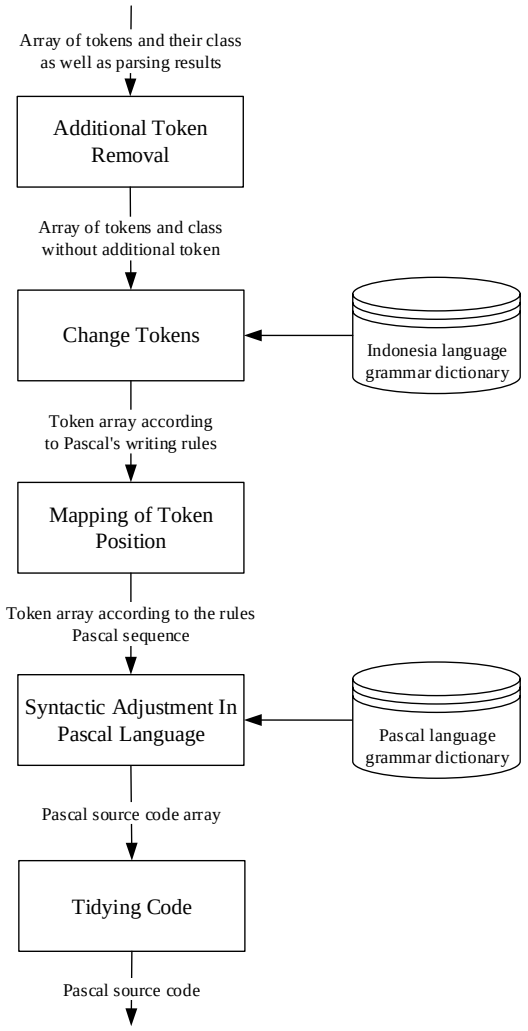

Figure 4 Code Generation Block Diagram

a. Additional Token Removal
Removal of additional token is the stage of deleting a token that has a 'AdditionalToken' class, because in this study the token is considered unnecessary.

Table 9 Example of Additional Token Removal Results

Sebelum	
Token	**Kelas**
buat	AdditionalToken
aplikasi	Keyword
tampil_string	IdentApp
.	Delimiter
tampilkan	Keyword
hallo world	String
.	Delimiter
Sesudah	
Token	**Kelas**
aplikasi	Keyword
tampil_string	IdentApp
.	Delimiter
tampilkan	Keyword
hallo world	String

.	Delimiter

b. Change Tokens
At the stage of changing tokens, tokens that have the class 'Keyword', 'Arithmetic Operator' and 'Number' will be changed according to the rules in Pascal language. This conversion utilizes Indonesian-made Grammar and *Pascal* Grammar.

Table 10 Example of Changing Token Results

Before	
Token	**Class**
aplikasi	Keyword
tampil_string	IdentApp
.	Delimiter
tampilkan	Keyword
hallo world	String
.	Delimiter
After	
program	Keyword
tampil_string	IdentApp
.	Delimiter
writeln	Keyword
hello world	String
.	Delimiter

c. Mapping of Token Position
Token position mapping is the stage of adjusting the order in which tokens appear according to the rules of the Pascal language. The token mapping in this study was carried out for the assignment command.

d. Syntactic Adjustment In Pascal Language
At this stage it is similar to the parsing process, except that it does not aim to check the order in which tokens appear, but to enter tokens token mapping results into Pascal language grammar, because there are some codes that do not exist in natural language writing, such as 'begin' end ', etc.

Table 11 Example of Syntactic Adjustment Results in Pascal Language

Before	**After**
program	program
tampil_string	tampil_string
.	;
writeln	begin
hello world	writeln
.	(
	‘
	hello world
	‘
	)
	;
	end.

e. Tidying Code
In the last stage, the token that has been inserted with the code needed in Pascal's language will be tidied so that it is easy to understand.

Tabel 12 Examples of Code Making Results

Before	**After**
program	program tampil_string ; begin writeln (‘hallo world’) ; end.
tampil_string	
;	
begin	
writeln	
(	
‘	
hello world	
‘	
)	
;	
end.	

4.7. Test Result Analysis

Accuracy testing is done by comparing the translation results obtained from the system with the expected results. Tests are performed on variable creation commands, calling readln functions, calling writeln functions, and basic arithmetic operations. Tests are carried out on 100 test data which are divided into 8 command combinations. The results of translational testing from Indonesian to source code in Pascal language can be seen in Table 13.

Table 13 Accuracy Test Results

Commands Combination	**Number of Test Data**	**Translation Results**	
		Right	**Wrong**
Without variables, 1 statement	10	9	1
Without variables, more than 1 statement	10	9	1
1 variable, 1 statement	10	10	0
1 variable, more than one statement	10	10	0
More than 1 variable, more than 1 statement	10	10	0
1 arithmetic operator	15	15	0
2 arithmetic operators	18	18	0

More than 2 arithmetic operators	17	17	0
Total	100	98	2

The accuracy value obtained in this study is 98%. Translation errors are caused by the system's scanning stage that cannot distinguish decimal fractions with two numbers separated by commas. Examples of wrong detection sentences can be seen in Table 14.

Tabel 14 Scanning Result Error

Input Text	Expected Results	Results of the system
tampilkan 1, 2, dan 3.	tampilkan	tampilkan
	1	**1,2**
	,	,
	2	dan
	,	3
	dan	.
	3	
	.	

In this study there are several cases that have not been addressed, and can be used as references in future studies. Cases that cannot be dealt with are as follows.

a. The system has not been able to distinguish decimal fractions with two numbers separated by commas, as in Table 14.
b. The input sentences are still limited because Indonesian grammar in this study is still simple. Examples of sentences that can be handled and cannot be seen in Table 15.

Table 15 Examples of Assignment Commands That Are Handled And Not Handled

Example of sentences	Status
Tambahkan 12 dengan 1 masukkan hasilnya ke hsl.	Handled
12 ditambah 1 masukkan hasilnya ke hsl.	Handled
Hsl diisi dengan hasil penjumlahan 12 dan 1.	Not yet handled
Jumlahkan 12, 1, dan 2 masukkan hasilnya ke hsl.	Not yet handled

c. The system has not been able to handle cases of branching and repetition, for example in Table 16.

Table 16 Example of Branching and Repetition Commands

Selection statement
Jika indeks tidak sama dengan e, maka tampilkan lulus, namun jika tidak tampilkan tidak lulus.
Repeat Statement
Selama i kurang dari 10, maka tampilkan i.

d. The system has not been able to analyze input data semantically, as in Table 17.

Table 17 Examples of Semantically Wrong Commands

Input Text	Information
Buat aplikasi uji_string. Buat variabel nama_depan dan nama_belakang dengan tipe data string. Baca nama_depan dan nama_belakang. Tampilkan hasil nama_depan dikali nama_belakang.	There is a command that multiplies two variable types of string data.
Buat program uji_coba. Buat variabel a dengan tipe data string. Kemudian 12 dimasukkan ke a.	There is a command to enter an integer value into a string variable.

e. The system has not been able to handle misspellings in the input text, for example in Table 18.

Table 18 Error Spell Example In Input Text

Input Text	Information
Buat aplikasi uji_string. Tampikan hallo world.	There is a misspelled word ‘tampikan’, should ‘tampilkan’.

5. CONCLUSION

Based on the tests that have been done, it can be concluded that the system can translate natural languages in Indonesian into source code in Pascal language. This study received an accuracy value of 98%.

There are several cases that have not been addressed in this study, therefore this research can be developed again in the future. There are suggestions that can be applied to further research, namely as follows.

a. Add rules that can distinguish fractions with two numbers separated by commas as in Table 4.14.
b. Add the complexity of Indonesian grammar and Pascal language to detect more diverse sentences.
c. Add translation features for the basic structure of branching and repetition.
d. Add semantic analysis features to the analysis phase of the Indonesian input text.
e. Add misspelled word correction features.

BIBLIOGRAPHY

[1] R. Munir, Algoritma dan Pemrograman, Bandung: Informatika, 2011.

[2] D. Satu dan A. Avinash, "Unrestricted Natural Language Implementation in Programming," *International Research Journal of Engineering and Technology*, vol. 03, no. 10, pp. 470-476, 2016.

[3] S. Nadkarni, P. Panchmatia, T. Kaewa dan S. Kurhade, "Semi Natural Language Algorthm to Programming Language Interpreteur," *International Conference on Advances in Human Machine Interaction*, 2016.

[4] T. Dirgahayu, S. N. Huda, Z. Zukhri dan C. I. Ratnasari, "Automatic Translation from Pseudocode to Source Code: A Conceptual-Metamodel Approach," *2017 IEEE International Conference on Cybernetics and Computational Intelligence, CyberneticsCOM 2017 - Proceedings*, pp. 122-128, 2018.

[5] W. Budiharto dan D. Suhartono, Artificial Intelligence, Yogyakarta: Andi Offset, 2014.

[6] M. Arhami, Konsep Kecerdasan Buatan, Yogyakarta: Andi Offset, 2006.

[7] K. K. Purnamasari dan I. S. Suwardi, "Rule-based Part of Speech Tagger for Indonesian Language," *IOP Conference Series: Materials Science and Engineerin*, vol. 407, pp. 1-4, 2018.

[8] E. Nugroho, Bahasa-Bahasa Pemrograman, Yogyakarta: Andi Offset, 1992.

[9] F. Utdirartotmo, Teknik Kompilasi, Yogyakarta: Graha Ilmu, 2005.

[10] M. Nazir, Metode Penelitian, Bogor: Ghalia Indonesia, 2014.

[11] R. S. Pressman, Software engineering : A Practitioner's Approach Edisi 7, New York: McGraw-Hill, 2010.

[12] Y. Bassil, "A Simulation Model for the Waterfall," *International Journal of Engineering & Technology*, vol. 2, no. 5, 2012.