GRAPHICAL LANGUAGE FOR MODELING ALGORITHMS

Dan VÎRTOSU[1],
Dina-Alexandrina BÎZGU[1],
Teodora POSTOVAN[2],
Dumitru COVAL[2],
Cristian DELINSCHI[2]

[1]*Technical University of Moldova, Faculty of Computers, Informatics and Microelectronics, FAF18, Chișinău, Republic of Moldova*

[2]*Technical University of Moldova, Faculty of Computers, Informatics and Microelectronics, FAF182, Chișinău, Republic of Moldova*

* Corresponding author: Dan Vîrtosu , dan.virtosu@isa.utm.md

Abstract. *In this paper it is analyzed what means domain specific language, which are the main advantages and how to develop a domain specific language for modeling algorithms, how to define its grammar and what set of rules are applied. The users of this domain specific language will be the kids between 4 and 7 years old. Due to this domain specific language, the users will be able to do some basic commands, such as moving and implement some procedures in order to reproduce an image, or to create one independently. The main purpose of this domain specific language will be the development of analytical skills, problem solving but not least developing of creativity.*

Keywords*: domain specific language, grammar, lexical analysis, programming language.*

Introduction

A domain specific language (DSL) is a programming language with a higher level of abstraction optimized for a specific class of problems. Using a DSL can bring benefits in term of: reduced time to market, productivity, quality assurance, development cost [2].

The main objective of the article is to develop a graphical language for modeling algorithms. This will involve other steps that need to be followed, such as: developing a graphical language, analysis of the DSL syntax, establishment of the grammar of the DSL, inurement 4-7 years old kids algorithmic thinking skills, teaching the 4-7 years old kids basic steps of programming [1].

It was worked on creation the platform that is able to teach kids by simple steps to develop algorithmic thinking skills. They will be able to use the drawing tools like moving the dinosaur to any direction, in this way leaving different lines in order to obtain a figure. For example, they may have an example figure image and they must draw it by moving dinosaur, and to select by itself the number of steps, the direction, type of line and degree of the angle of moving it [4].

With developed app, kids would can learn the planning of complex tasks using simple elements. An important skill is reusing previous work. Children also would can learn how to use graphics and space in coding. The important ideas of commands, procedures, variables, loops, and conditionals are introduced. The proposed language encourages children to develop their own ideas and use their imagination [6].

The created application will teach kids the basic points of programming, such as:

1. Definition of the program.
2. Planning a solution.
3. Program coding.
4. Testing the program.
5. Surprise that the program works.

Reference grammar

A proposed context free grammar *G* is an ordered quadruple ***G*=(V_N, V_T, *P*, *S*)** where:

1) V_N - is a finite set of non-terminal symbols.
2) V_T- is a finite set of terminal symbols.

$$V_N \cap V_T\text{-} = \emptyset$$

3) *S* - is a start symbol.
4) *P* – is a finite set of productions of rules.

Grammar of a programming language is a highly technical way of describing a set of formal rules that govern how the programming language is constructed and present the valid tokens or lexemes. According to a valid grammar, the code of any programming language can be easily implemented without any errors and troubles. In the following table can been seen the guide of our grammar. On the left side there are symbols and on the right side we can see the significance of each symbol [3].

Table 1

Meta-notation

<foo>	means foo is a nonterminal symbol
foo	(in **bold** font) means foo is a terminal symbol
x *	means zero or more occurences of x
x +	separated list of one or more x's
|	separates alternatives

Next it will be presented the terminal/non-terminal symbols and the production offered by the development DSL:

● V_T ={ start, stop, inainte, stanga, dreapta, inapoi, poz_xy, sare, linie_dreapta, linie_intrerupta, linie_punctata, seteaza_pozitia_initiala, ciclu_de_repetare, comanda_de_asteptare, [a-z,A-Z], [0-9], negru, verde, rosu, albastru, galben, roz, orange, maro, violet, azuriu, bej };

● V_N ={<program>, <comanda>, <chemare procedura>, <declarare procedura>, <nume>, <parametru>, <expresie>, <nume>, <linie>, <culoare>, <litere>, <sare>, <poz_initiala>, <repeta>}

● **P=**{

<program> →**start** <comanda> | <declarare_procedura> **stop**
<comanda>→ **inainte | stanga | dreapta | inapoi | poz_xy** | <chemare_procedura> |
| sare | poz_initiala | repeta | **asteapta** | <linie> |
<chemare procedura > → <nume> <expresie>*
<declarare procedura>→ <nume> <parametru>*
<parametru> → <nume> (' , ' <parametru>)*
<repeta>→ 'repeta' <numar >
<nume>→ <litere>
<poz_xy> → **poz_xy** <expresie>< expresie>
<sare> → **sare** <expresie>
<start> → **start**
<expresie > → <numar>+ |<culoare>
<stop> → **stop**
<linie> → **linie_dreapta** <expresie> | **linie_intrerupta** <expresie> |**linie_punctata** <expresie> |
<seteaza_pozitia_initiala>→ **poz_initiala** <expresie>< expresie>
<ciclu_de_repetare >→ **repeta<** expresie>
<comanda_de_asteptare>→ **asteapta** <expresie>
<litere>→ **[a-zA-Z]**
<numar>→ **[0-9]**
<culoare >→ **negru | verde | rosu | albastru | galben | roz | orange | maro | violet | azuriu | bej**

Additional Rules

These sorts of rules add additional constraints for the validation of DSL program. A program that follows all the rules of a grammar and does not violate any of these constraints is said to be legal one:

1) Every program should begin with the keyword **start** and finish the execution with the keyword **stop**.
2) No need for parentheses for command execution, procedure declaration and procedure invocation.
3) Calling the repetitive procedure **repeat** only the following line will be executed by this procedure, others will be executed in a normal way.
4) Numbers from **numar** non-terminal symbols should be greater or equal to 0.

Table 2

Lexical Implementation

token value	**type**
start, stop	keyword comand
inainte | stanga | dreapta | inapoi | poz_xy | chemare_procedura | | sare | poz_initiala | repeta | asteapta | linie	keyword
[0-9]+	number
,	digit

Table 3

Program Result

Input	**Output**
start sare 2 dreapta 3 inainte 5 cerc 5 dreptunghi 4 , 5 repeta 3 asteapta 1 stop	value: 'start' , type: 'keyword comand' value: 'sare' , type: 'keyword' value: '2' , type: 'numar' value: 'dreapta' , type: 'keyword' value: '3' , type: 'numar' value: 'inainte' , type: 'keyword' value: '5' , type: 'numar' value: 'cerc' , type: 'keyword' value: '5' , type: 'numar' value: 'dreptunghi' , type: 'keyword' value: '4' , type: 'numar' value: ',' , type: 'digit' value: '5' , type: 'numar' value: 'repeta' , type: 'keyword' value: '3' , type: 'numar' value: 'asteapta' , type: 'keyword' value: '1' , type: 'numar' value: 'stop' , type: 'keyword comand'

Conclusion

As a consequence of this research we have come to the conclusion that coding for kids is growing in popularity, as many families view computing as a new literacy that will be as important as math and science in tomorrow's job market. One of the most important piece of advice was to implement the application that permit the codding to be funny, because coding for kids doesn't need to be boring.

Such a programming language will only bring benefits to kids and their parents and first of all they will use the time more efficiently, both the time of the child, in order to improve some skills as well as the parents, that will invest in the process of growing of their kids.

References

1. VOLTER M. *DSL Engineering. Designing, Implementing and Using Domain specific language*s, http://dslbook.org, 2010-2013
2. https://tomassetti.me/domain-specific-languages/ - Introduction to DSL
3. https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-035-computer-language-engineering-spring-2010/ - Reference grammar
4. https://www.idtech.com/blog/choose-best-programming-language-your-child
5. https://www.tynker.com/blog/articles/ideas-and-tips/how-coding-helps-kids-develop-key-21st-century-skills/
6. https://howtospell.co.uk/benefits-of-coding-for-kids