DOMAIN SPECIFIC LANGUAGE FOR DOCUMENT EDITING

Constantina GÎLCA*, Cristian IONEL, Cristian-Sergiu TAFUNE, Silviu LOZOVANU, Victor BOTNARU

Department of Software Engineering and Automatics, FAF 202, Faculty of Computers, Informatics and Microelectronics, Technical University of Moldova, Chișinău, Moldova

*Correspondent author: Gîlca Constantina, constantina.gilca@isa.utm.md

Abstract. *A Domain Specific Language (DSL) for reducing editing time of documents is presented. The domain which it addresses is the academic circle. It was reported that researchers spend up to twice more time on editing a document than writing its content. Ergo, this paper aims to present an efficient solution and explain the grammar and the functionalities behind the DSL in question. This language gives a new approach to the document creation – it adapts the template to the text automatically.*

Key words*: research, documents, DSL, editing, grammar*

Introduction

A Domain Specific Language is a programming language with a higher level of abstraction optimized for a specific class of problems [1]. Since this paper aims to present a solution for minimizing the time consumption on document editing, this DSL's purpose is to adapt the content of the paper to a set of specified rules for document representation and to ease the process of tracking references and contents for the table at the third page.

For the former one, the DSL has some strict defined rules to which it edits the text. For example, in a classic university report, each chapter's title has to be written in uppercases, bold, with the Times New Roman font and 14px size. Those are details that might be overlooked by the user, but the DSL adapts the text to the rule.

For the latter one, it has been proven of much difficult to track the references used throughout the research and of much boredom to track the pages at which each chapter and subchapter starts. Hence, the DSL has some predefined functions, where at each new chapter or subchapter defined it is automatically added to the table of contents with the corresponding page, and each reference is added upon declaration to the list and the text is completed with its corresponded number.

Therefore, the code introduced by the user is nothing more than instructions and blocks of texts that the DSL uses to generate a document by, written with respect to all the formatting rules in less than minutes.

1. Grammar

The syntax of a programming language is the set of rules that define which arrangements of symbols comprise structurally legal programs. Grammar is defined by four elements in an order of $G = (V_N, V_T, P, S)$. The meaning of the elements is as follows:

- V_N - set of nonterminal symbols.
- V_T - set of tokens or terminal symbols.
- P - set of production rules.
- S - start symbol.

For further understanding, in the Tab. 1 are listed the meta notations used throughout this paper.

Table 1

Meta Notations

Symbol	Meaning
<abc>	A nonterminal symbol
abc	A terminal symbol
x^*	Zero or more occurrences of x
x^+	One or more occurrences of x
$x^?$	Zero or one occurrence of x
|	Separates alternatives

For the project in question, the elements were defined as follows.

V_N = { <call_method>, <method_body>, <method_body_string>, <method_name>, <method_parameter>, <begin_method>, <end_method>, <parameter>, <f_name_parameter>, <string_parameter>, <extension_parameter>, <number_parameter>, <image_parameter>, <link_parameter>, <type_doc_parameter>, <text>, <text_char>, <Lcase_letter>, <Ucase_letter>, <number>, <img_extension_name>, <symbol>, <round_bracket_Left>, <round_bracket_Right>, <colon>, <comma>, <low_line>, <quotation_mark>, }

V_T = {[**0-9**], [**a-z**], [**A-Z**], **({ }) : " , _ ? / - . " " {^ ^}** , **report**, **research**, **docx**, **pdf**, **jpeg**, **png**, **jpg, ε**}

S = {<call_method>}

P = {<call_method> → <method_name> <round_bracket_Left> <method_parameter>*
<round_bracket_Right>
| <method_name> <round_bracket_Left> <method_parameter>*
<round_bracket_Right> <colon> <begin_method> <method_body>
<end_method>
<method_body> → <text>+ <call_method>*
<method_name> → <Lcase_letter>+ | <method_name> <low_line> <method_name>
<method_parameter> → <parameter> | <parameter> <comma> <method_parameter>
<parameter> → <f_name_parameter> | <string_parameter> |
<extension_parameter> | <number_parameter> |
<image_parameter> | <link_parameter> | <type_doc_parameter>
<f_name_parameter> → <text_char>+ <number>*
<string_parameter> → <quotation_mark> <text>+ <quotation_mark>
<extension_parameter>→ **pdf** | **docx**
<number_parameter> → <number>+
<image_parameter> → <string_parameter> **,** <text> **.** <img_extension_name>
<link_parameter> → <text>
<type_doc_parameter> → **report** | **research**
<text> → <text_char>+ <number>* <symbol>* <text>*
<text_char> → **a** | **b** | … | **z** | **A** | **B** | … | **Z** | **ε**
<Lcase_letter> → **a** | **b** | … | **z**
<Ucase_letter> → **A** | **B** | … | **Z**
<number> → **0** | **1** | … | **9**
<doc_extension_name> → **docx** | **pdf**
<img_extension_name> → **jpeg** | **jpg** | **png**
<begin_method> → **{^**
<end_method> → **^}**
<symbol> → **(** | **{** | **}** | **)** | **:** | **"** | **,** | **_** | **?** | **/** | **-** | **.**
<round_bracket_Left> → **(**
<round_bracket_Right> → **)**
<comma> → **,**
<colon> → **:**
<low_line> → _
<quotation_mark> → **“** | **”**
}

2. Derivation tree

Derivation tree is a graphical representation for the derivation of the given production rules of the context free grammar (CFG). It is a way to show how the derivation can be done to obtain some string from a given set of production rules [2].

Hence, an example of derivation tree is used with the purpose of a better understanding of how the grammar works in generating correct instructions. Considering Fig. 1 it can be seen how choosing repeatedly some rules from the set P generates a valid instruction for defining a new chapter.

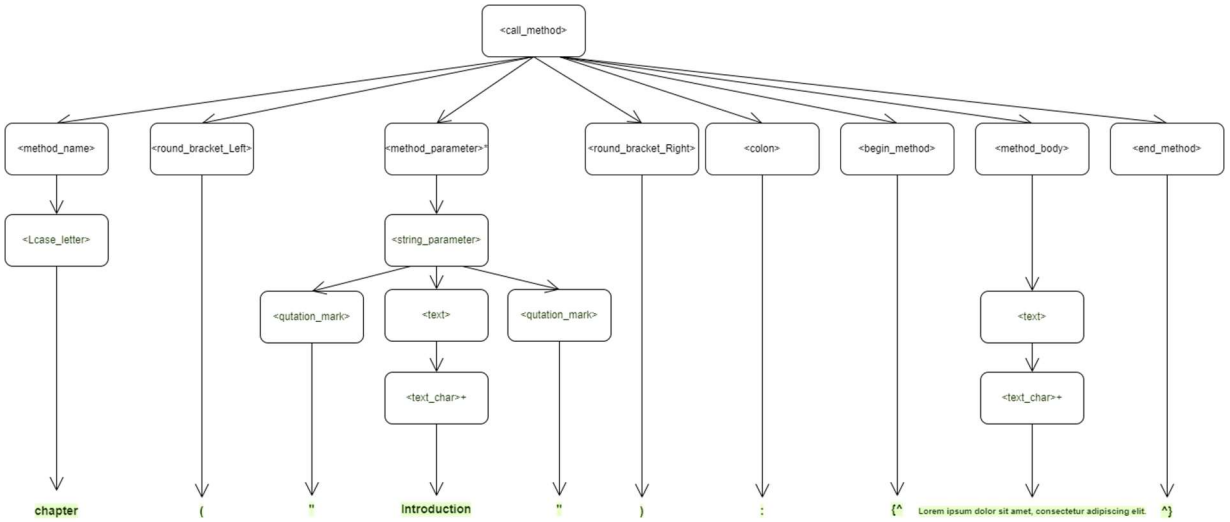

Figure 1. Derivation tree example

3. Practice example

For a better perspective of how this project will work, it was considered the following code example:

```
document( report, docx)
title("Report Example")
subject("Formal Languages")
author("Gîlca Constantina")

table_of_contents(default)

chapter("Introduction"):
{^
      Lorem ipsum dolor sit amet, consectetur adipiscing elit. In a  nisl
enim. Ut semper, velit hendrerit gravida volutpat, odio neque  malesuada orci, et
imperdiet mi augue in dolor. Etiam efficitur      ultricies risus nec posuere.
Vestibulum accumsan venenatis mauris ac     tincidunt. Mauris ut massa     quam.

      subchapter("Problem Analysis"):
      {^
            Ut condimentum dignissim augue, at bibendum nunc blandit
      eu. Vivamus augue mauris, scelerisque et venenatis et, tincidunt ut  nisi.
Suspendisse interdum massa ut porta condimentum. Proin lorem  nibh, pretium at diam
et, ultrices semper arcu. Donec accumsan dolor    enim, ac luctus mi
      fringilla ut. Integer accumsan lectus accumsan    semper aliquam. Suspendisse
scelerisque sem vitae libero fermentum, in  consectetur enim sollicitudin. Duis
vestibulum gravida augue, eu   rhoncus diam pulvinar a.
```

```
                table_name("Statistics")
                table_row("Year", "Company", "Mean")
                table_row("2001", "Sony", "4.1")

                image(“Face”,/face.png)

                list("For example", arabic_numbers):
                {^
                      item("Item 1")
                      item("Item 2")
                      item("Item 3")
                ^}
          ^}
^}
```

After its compilation, the DSL generated the file in the Fig. 2, which represents the title page, the contents table and the other content generated by the user.

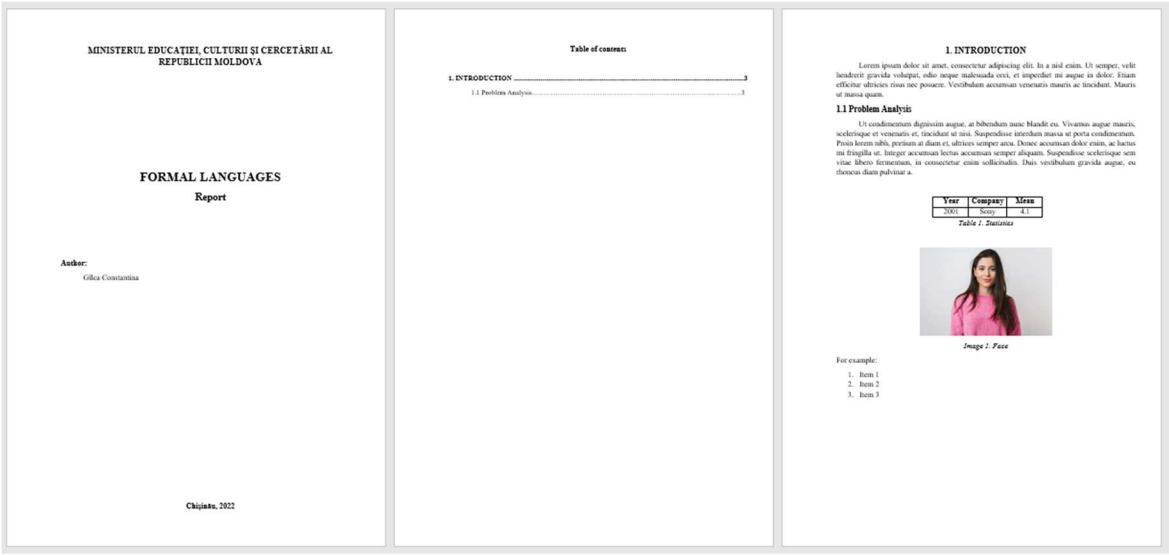

Figure 2. Generated document by DSL

Conclusions

The process of gathering information, implementing projects and writing the results is a joyful activity, full of surprises and knowledge. But it ends in an unnecessary lose of time on document editing and page counting to fit a set of rules defined by others. Moreover, to assure the acceptance of the paper in scientific communities, the problem becomes not as much its content as the way of serving it in terms of order, colours, size and organization.

To ease the activity of the academic circle and to assure the researchers with more time on writing than on editing, the solution above was presented. Its simplicity and efficiency assure easy use for any person and in any domain of consideration.

In the end, DSL is a tool to deliver great content in the right way, easy to do and easy to understand.

References

1. Domain-Specific Languages, JET BRAINS. [online]. [accessed 01.03.2022]. Accessible at: https://www.jetbrains.com/mps/concepts/domain-specific-languages/
2. What is a Derivation tree in TOC?, tutorialspoint. [online]. [accessed 01.03.2022]. Accessible at: https://www.tutorialspoint.com/what-is-a-derivation-tree-in-toc